ára: 290 Ft

infopen

nyílt rendszerek magazinja

V. évf. 6. szám 1997. június



A hónap Internet-ajánlata: HLE

- A Web cache-elése
- Objektumrelációs adatbázis-kezelők
- A Jáva és a HTTP protokoll
- TOPszáz topLÁZ
- Könyvajánlatok

„Technológiai innováció, termékforgalmazás és kulcsrakész alkalmazásfejlesztés”



Dömölki Bálint, az IQSOFT igazgatótanácsának elnöke és Szabó Tamás ügyvezető igazgató

infopen®

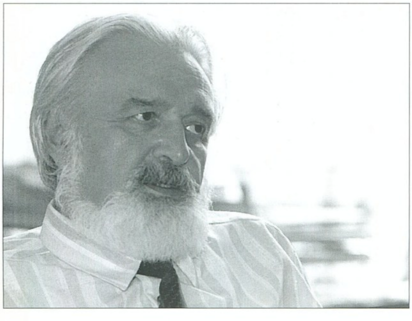
Nyílt rendszerek magyarországi hírmagazinja
Kiadja az **OpenInfo Kiadó**
Felelős kiadó: **Dr. Vas Zoltán**
Alapító főszerkesztő: **Kovács Attila**
Szerkesztőbizottság:
Dr. Demetrovics János, Nagy Miklós, Dr. Remzső Tibor, Dr. Sima Dezső, Dr. Telbisz Ferenc
Főszerkesztő: **Dr. Hutter Ottó**
Olvasószerkesztő: **Gams Judit**
Művészeti és műszaki vezető (fotó): **Szabó Tibor**
Titkárságvezető: **Polyák Erzsébet**
Nyomás és kötés: **Akadémiai Nyomda**
Felelős vezető: **Freier László**
Levilágítás: **LaserGraph**
A cikkekben és táblázatokban szereplő adatokat gondosan ellenőrizzük. Az esetleg mégis előforduló pontatlanságokért és tévedésekért azonban a kiadó nem vállal felelősséget.
Előfizetés:
az **OpenInfo** kiadónál
egy évre: 1900 Ft + áfa
Telefon: 166-5644/447, 413;
fax: 166-7503;
postacím: 1111 Budapest, Kende u. 13.
Internet címek: infopen@ind.eunet.hu,
http://www.eunet.hu/infopen
Hirdetésfelvétel:
Pap Katalin, Árvai Katalin
Tel.: 214-9492, 156-3211/168, 200 Fax: 214-9492, 156-3211/201
E-mail: alaplap@mail.datanet.hu

HU ISSN 1217-1905

Kutatóintézetből szoftverház

Technológiai innováció, termékforgalmazás és kulcsrakész alkalmazásfejlesztés

Fennállásának nyolcadik évében jelentős mérföldkőhöz érkezett az IQSOFT. Az 1990-ben egy nagy múltú kutatóintézet, az SzKI egyik laboratóriumából kb. negyven fővel alakult részvénytársaságnál jelenleg több mint hetven állandó munkatárs dolgozik, és árbevételük meghaladja a félmilliárd forintot. Bár nem szakadtak el a kutatóintézeti gyökerektől, hiszen a relációs adatbáziskezelés vagy az objektumorientált és más fejlett technológiákkal kapcsolatban a piac meghatározó szereplőivé váltak, az IQSOFT mára késztermékek egyre szélesedő skáláját forgalmazva és számos nagy, egyedi alkalmazói rendszert fejlesztve az ország egyik élenjáró szoftverházává vált. Az alábbi interjúban *Dömölki Bálint*, az igazgatótanács elnöke és *Szabó Tamás* ügyvezető igazgató adnak képet a cég fejlődéséről, a fejlesztési stratégiáról és a bevételek szempontjából kulcsfontosságú tevékenységekről.



Dömölki Bálint, az igazgatótanács elnöke

Az első nyolc év meglehetősen szorosan kapcsolódott Dömölki Bálint nevéhez, hiszen mindvégig ő irányította a cég operatív tevékenységét, és ő állt a stratégiát meghatározó igazgatóság élén is. Kezdjük hát a beszélgetést a közelmúlt szervezeti és személyi változásaival: mi motiválta a cégvezetés átalakítását, és hogyan oszlanak meg a feladatok az ügyvezető, az igazgatóság és az elnök között a jövőben?

D. B.: Az IQSOFT növekedése egy stabil, de mérsékelt ütemű fejlődési szakasz után az utóbbi időszakban jelentősen felgyorsult. Az 1996-os árbevétel kb. 60%-kal haladta meg az előző évit, és ezzel átlépte a félmilliárd forintot. Ehhez 26 millió forintos nyereség társult, és a kezdeti negyvenről az alkalmazottak száma hetven fölé emelkedett. Ennek a felgyorsult növekedési ütemnek a megtartása óhatatlanul megkövetelte a menedzsment megerősítését, ami azt jelentette, hogy szétválasztottuk az ügyvezetői és az elnöki posztot, valamint újabb vezetőket neveztünk ki az igazgatóságba. Számunkra mindig lényeges volt, hogy műszaki/technológiai téren élen járjunk, ezért olyan fiatal, energikus vezetőt kerestünk az operatív irányítás átvételére, aki otthonosan mozog a korszerű technológiákban. Úgy érzem, hogy Szabó Tamás személye — aki az elmúlt másfél évben az adatbázis-kezeléssel és az objektumorientált technológiákkal foglalkozó osztály irányítója volt — garancia arra, hogy a cég szakmai irányvonala változatlan maradjon. Ugyanakkor, mivel rálátása van kereskedelmi tevékenységünkre is, a kereskedelmi és marketing-irányultságunk megerősítésében is számíthatunk munkájára, mint ahogy a teljes operatív vezetés javaslataira is építhetünk a hosszú távú stratégia megtervezésében.

Az igazgatótanács elnökeként én a jövőben kifejezetten a stratégiai kérdésekre szeretnék koncentrálni, amiben nagy segítséget fog jelenteni, hogy az igazgatótanácsot kibővítettük *Sipka Júlia* marketing- és kereskedelmi, valamint *Langer Tamás* műszaki igazgatóval. Természetesen nemcsak a felső vezetést erősítettük meg, hanem folyamatosan bővítjük szakembergárdánkat is. Legutóbb például objektumorientált és adatbázis-technológiákkal foglalkozó csapatunk létszáma gyarapodott hat elismert szakemberrel, köztük *Kovács Andrással*, aki Szabó Tamás helyét vette át az Adatbázis-fejlesztési és értékesítési osztály élén.

Már eddig is többször szóba kerültek az IQSOFT kutatóintézeti gyökerei és technológiai irányultsága. Konkrétan mely műszaki területek állnak a cég kutatási, fejlesztési, illetve kereskedelmi tevékenységének a középpontjában?

D. B.: Ami a kutatóintézeti gyökereket illeti, folyamatosan dolgozik egy néhány fős csoportunk olyan, az Európai Unió által finanszírozott kutatási projekteken, amelyek korábban logikai programozási, adatbáziskezelő, újabban pedig objektumorientált eszközök kipróbálását és hazai bevezetését célozzák. A cég fennállásának első három évében — amikor az országban még a clipperes, illetve dBase-alapú alkalmazások uralták a piacot — a korszerű, második generációs adatbázis-kezelő rendszerek meghonosítására koncentráltunk, ami Oracle termékek forgalmazását, illetve Oracle-alapú alkalmazások fejlesztését jelentette. Azt hiszem, túlzás nélkül mondhatom, hogy az IQSOFT-nak meghatározó szerepe volt abban, hogy az Oracle Magyarországon még a világpiaci átlagot is meghaladó pozíciókat szerzett. Mint ismeretes, 1993-ban az Oracle úgy döntött, hogy saját leányvállalattal jelenik meg az országban, és ez természetesen számunkra is új helyzetet teremtett. Fejlesztéseink során ugyan továbbra is stratégiai platformként támaszkodunk az Oracle adatbázis-kezelő termékeire, de nyitottunk egyrészt más, hordozhatóságot garantáló 4GL és CASE fejlesztőeszközök felé, másrészt komplett megoldások szállításába fogtunk különböző vertikális alkalmazási területek számára (pl. integrált irodai rendszerek, könyvtári rendszerek, termelésirányítás).

Sz. T.: Ahogy annak idején a relációs adatbázis-kezelés területén, úgy az elmúlt években az objektumorientált technológiák terén próbál egyfajta úttörő szerepet betölteni az IQSOFT. Magyarországon körülbelül két évvel ezelőtt, az elsők között kezdtünk el CORBA fejlesztőeszközökkel dolgozni. Ezek a befektetések éppen mostanában kezdenek beérni, amikor a nagy adatbázis-kezelő rendszerek új verzióiba fokozatosan beépülve a relációs adatkezelés mellett lassan általánosan elérhetővé válik az objektumalapú is. A sokéves előkészületeknek köszönhetően nálunk az adatbázis-tervezők és a programozók ugyanazt a nyelvet beszélik, úgyhogy minden jel szerint az elsők között jelenhetünk majd meg Magyarországon éles, üzleti célú, CORBA-alapú objektumorientált alkalmazói rendszerekkel.

Térjünk most át a technológiáról a termékszintű kereskedelmi tevékenységre. Mekkora a termékeladásból származó bevételek aránya az IQSOFT forgalmában, és milyen ennek a megoszlása termékcsaládokra vetítve?

D. B.: Természetesen mind a termékszintű kereskedelem, mind az egyedi alkalmazásfejlesztés arra a technológiai innovációra épül, amelyről eddig beszéltünk. Ami az arányokat illeti, bevételeinknek kb. az egyharmada az IQ-Best (IQ Business Solution Technologies) fan-

tázianévre keresztelt termékcsoport forgalmazásából, illetve az ehhez kapcsolódó tanácsadói-fejlesztői tevékenységből származik. Idetartoznak mindenekelőtt az Oracle adatbázis-kezelő termékek, a Centura (korábbi nevén Gupta) adatbázis-kezelő és 4GL fejlesztőeszköz-készlet, valamint a Paradigm Plus CASE eszköz. Ezek a fejlesztőeszközök nemcsak Oracle-alapú, hanem teljesen hordozható alkalmazások készítésére is alkalmasak. A termékkör természetesen fokozatosan bővül, például újabban előtérbe kerülnek a különböző tesztelő- és teljesítmény-monitorozó, illetve -hangolóeszközök, továbbá az objektumgyűjtemények. Külön kiemelném például a PLATINUM DBVision adatbázis-monitorozó és ServerVision Unix szerver-monitorozó szoftvereket, melyek nagyvállalati alkalmazói rendszerek teljesítményhangolása során felbecsülhetetlen segítséget nyújtanak a rendszergazdáknak.

Sz. T.: Az IQ-Best termékcsalád két részből áll. A forgalom mintegy 90%-át a már említett, jól bevezetett, hagyományosnak mondható termékek adják. Ám ezek mellett mindig van néhány, a legújabb, legkorszerűbb technológiákat képviselő új termékünk is. A közelmúltban lettünk például disztribútorai az IONA Technologies Orbix termékcsaládjának, amely piacvezető a CORBA-alapú fejlesztőeszközök területén. De idesorolnám az ObjectStore objektumorientált adatbázis-kezelőt is, amely C++, Java és ActiveX elemek tárolását biztosítja, és a nagy teljesítményű DBMS-t igénylő objektumorientált alkalmazások motorja.

Ezek az eszközök mindenekelőtt hatalmas adatbázisokkal dolgozó Internet/intranet alkalmazások fejlesztéséhez teremtenek kitűnő alapot. Új termékünk a Persistence Software Persistence objektumrelációs csatolóeszköze is, amely szintén kategóriájának a legjobbja. Jó példája annak, hogyan ötvözhető az objektumorientált technológiából adódó gyorsabb adatfeldolgozás és navigálási képesség a kiforrott, elterjedt relációs adatbázis-kezelőkkel történő objektumtárolással.

A szoftvertermékek forgalmazása és a fejlesztési tevékenység mellett olyan komplett alkalmazói csomagok is megjelentek az IQSOFT kínálatában, mint az integrált irodai rendszerek, az OLIB könyvtárautomatizálási csomag vagy az IFS/Avalon integrált vállalatirányítási rendszer. Milyen súlya van ennek a tevékenységnek a cég hosszú távú stratégiájában?

D. B.: Néhány olyan alkalmazói termékcsalád is szerepel kínálatunkban, amelyekben megvan a bővülés lehetősége, és nagyon bízunk benne, hogy ezek előbb-utóbb komoly piaci részesedést vívnak ki maguknak. Azt a stratégiát választottuk, hogy nem egy-egy gazdasági ágazatra koncentrálunk, hanem viszonylag kiegyensúlyozott módon jelen akarunk lenni mind a banki, ipari, telekommunikációs, mind a kormányzati szektorban. Ez persze sokirányú speciális szakismeretet és alkalmanként tetemes kapacitásokat igényel, amit nem kizárólag saját alkalmazottakkal, hanem egy húsz-harminc fős stabil gárdát jelentő partnerhálózattal fedünk le. Ami a konkrét területeket illeti, legrégebben talán az integrált irodai rendszerekkel foglalkozunk, mivel rendszerintegrátorként már 1991-ben megkezdtük a magyar Hypermédia Systems Kft. által kifejlesztett Archiware termékcsalád különféle moduljainak a forgalmazását és rendszerbe integrálását DOKTÁR néven. Komplett munkafolyamatok, a hivatali ügymenet automatizálásához felvettük palettánkra a CSE WorkFlow-t, amelyet gyakran integrálunk más rendszerekkel, legsűrűbben éppen a DOKTÁR archiváló moduljával. Ebbe a körbe sorolható saját fejlesztésű iktatórendszerünk vagy az az optikai képfeldolgozó technológiánk, amely a Magyar Posta részére készített nagyon sikeres Napi Elszámolási Rendszer tapasztalataira épül. Immár két éve az ország összes postahivatalának napi elszámolási bizonylatait ez a NER nevű rendszer dolgozza fel.

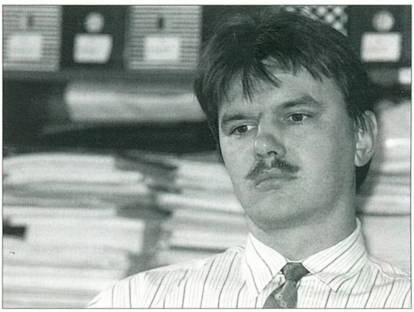
A könyvtárautomatizálás területén — annak ellenére, hogy több versenytárs is van ezen a nem túl nagy piacon — szintén szép sikereket értünk el az OLIB (korábbi nevén Oracle Libraries) rendszerrel. Ez tetszőleges típusú dokumentum katalogizálásától a kölcsönzésig minden könyvtári tevékenységet magában foglal, beleértve a pénzügyi és rendelési folyamatok felügyeletét, és támogatja a multimédia alkalmazását is, valamint az Interneten keresztüli elérést, lekérdezést, hipertextes keresést.

1993 táján döntöttünk úgy, hogy belevágunk az igazán éles üzemnek számító termelésirányításba, hogy összegyűlt fejlesztési tapasztalatainkat ipari körülmények között is megpróbáljuk hasznosítani. Oracle-kötődéseink következtében olyan terméket kerestünk, amely erre a platformra épül, így jutottunk el az Avalon rendszerhez. Külön szerencsénk, hogy időközben a svéd IFS cég lett a termék tulajdonosa, amelyik szintén az általunk is jól ismert Oracle/Gupta fejlesztési környezetet használja. Úgy látjuk, hogy az Üzleti Folyamatok Újratervezése (BPR) révén az integrált vállalatirányítási rendszerek piaca komoly fellendülés előtt áll Magyarországon is, ezért nagy reményeket fűzünk ehhez az üzletágunkhoz.

Végezetül szaladjunk egy kicsit előre az időben: milyen nagyobb projektek sikeres lezárása várható a közeljövőben?

Sz. T.: Annak ellenére, hogy projektjeink — mint a fenti példákból is látható — rendkívül széles tevékenységi kört ölelnek fel, arra törekszünk, hogy ne aprózzuk szét erőforrásainkat. Az IQSOFT árbevételének mintegy 80 százaléka 10-12 nagy ügyfélhez kapcsolódik, így elsősorban abban bízunk, hogy ezeket a nagylélegzetű projekteket folyamatosan tovább tudjuk vinni. Ezeknél a cégeknél általában komplex fejlesztői és tanácsadói tevékenység is társul a termékértékesítéshez.

Ízelítőül csak néhányat mondok a nagyobb projektjeink közül a már említett postai bizonylat-feldolgozó rendszeren kívül: a Westel 900 GSM-nél részt veszünk egy komplex adatfeldolgozó és számlázórendszer kidolgozásában, CSE WorkFlow-alapú dokumentumkezelő rendszert telepítünk a MOL Rt.-ben, komplett OLIB-alapú könyvtári rendszeren és több más alkalmazáson dolgozunk a Matávnál, és különböző speciális pénzügyi alkalmazásokat fejlesztünk a Magyar Nemzeti Bank számára. Ez utóbbiaknak külön érdekessége, hogy a fejlesztés során a legújabb CORBA/Java technológiát alkalmazzuk egy ilyen kritikus környezetben. És hogy az internetes fejlesztések területéről is mondjak egy példát: mi készítjük a nagyon ígéretes Me-



Szabó Tamás
ügyvezető
igazgató

gabolt projekt komplett informatikai hátterét. Ez a hagyományos elektronikus áruházakkal szemben olyan átfogó termékkatalógushoz hasonlítható, amely a felhasználó számára segít megtalálni, hogy egy-egy árucikk éppen hol a legolcsóbb.

Ennek a rendszernek a megvalósítása során szinte mindent „bedobunk" a jövő ígéretes technológiai megoldásaiból: az Object Designt, az ObjectStore objektumorientált adatbázis-kezelőjét használjuk, az alkalmazói modulokat C++-ban, illetve Javában fejlesztjük, és az elosztott hálózati architektúra az IONA Orbix nevű CORBA implementációjára épül.

HUTTER OTTÓ

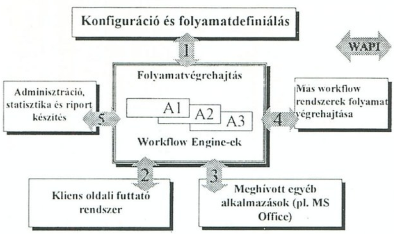
CSE/WorkFlow-val vezérelt munkafolyamat

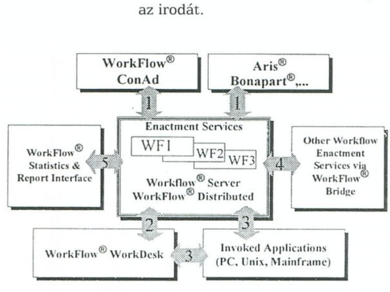
Tetszőlegeset tökéletesen

Az irodaautomatizálás eleinte csak egymástól elszigetelt tevékenységek gépesítésére koncentrált, s ezzel kihasználatlanul hagyott számos, a számítógépes rendszerek alkalmazásában rejlő lehetőséget. Napjainkban azonban — különösen nagyobb szervezeteknél — egyre meghatározóbb szerephez jut a munkafolyamat-vezérlés is.

A korszerű irodaautomatizálási szoftverek alkalmazási szempontból négy fő területet támogatnak: a dokumentumkezelést, az elektronikus levelezést, a munkafolyamatok modellezését és átszervezését. Közülük a munkafolyamatok kézben tartásának gépesítése hozott új szemléletet ezen a területen, s az ilyen szemlélettel készült termékek bizonyos értelemben valóban „automatizálják" az irodát.

A WfMC referenciamodell





A CSE Systems WorkFlow modellje – könnyen megfeleltethető a szabványnak

Workflow vagy workgroup?

A munkafolyamatok nyomon követését támogató szoftverek (workflow) a csoportmunkát támogató szoftverek (workgroup) világából érkeztek. Már az első hálózati alkalmazások is az utóbbi kategóriába tartoztak. Ezek lényege, hogy a számítógéppel dolgozó munkatársak közös erőforrásokat használtak például közös háttértár, nyomtató, adatbázis vagy egyéb információk formájában, de idetartoznak az elektronikus levelező- és iktatórendszerek is. Ebből következően a workgroup-alapú nyilvántartó, iktatórendszerek információközpontúak. Működése során a rendszer passzív, a felhasználó pedig aktív — a workgroup rendszer nem mondja meg a felhasználónak, hogy mit csináljon, csupán a közös erőforrásokat kínálja fel.

A workflow új szemlélettel közelít a munkafolyamatokhoz, túllép a workgroup filozófiáján. Lehetővé teszi, hogy modellezzük az elemi tevékenységek láncolatát alkotó munkafolyamatokat, és keretek között tartja a munkatársak tevékenységét. Egy elektronikus levelezőrendszerben a felhasználó oda küld üzenetet, ahová akar, egy workflow rendszerben viszont csak oda küldhet, ahová az adott munkafolyamat szempontjából a legcélszerűbb. Természetesen az alkalmazónak itt is van egyfajta döntési szabadsága, kialakítható olyan modell, amelyben több helyre is küldhet üzenetet, de ez minden esetben korlátozottabb, mint egy workgroup rendszerben. A workflow esetében tehát a rendszer dönti el, hogy mit tehetnek az egyes felhasználók, vagyis a rendszer aktív, a felhasználó pedig — a rendszerhez viszonyítva — passzív (a hálózati erőforrások transzparens elérése persze itt is biztosított); alkalmazása a munkatársak szintjén segíti a munkafolyamatok lebonyolítását, a vezetés számára pedig a gombnyomásra történő áttekinthetőség, a kontroll lehetőségét adja.

Workflow rendszereket ott érdemes bevezetni, ahol a munkafolyamatok jól strukturáltak, azaz megfelelő mélységben leírhatók, és általában mindig egy „dramaturgia" szerint mennek végbe; egy számlafeldolgozás például jól algoritmizálható. Szoftverfejlesztés esetén mindez már nem mondható el, hiszen ez a tevékenység tipikusan nem előre modellezhető folyamat, itt tehát nem lenne hatékony a workflow rendszer alkalmazása.

Szabvány születik

Több cég is ajánl workflow filozófiát követő rendszert, némelyik azonban még jól érzékelhetően hordozza a workgroup szemléletet, nagyrészt annak köszönhetően, hogy már meglévő workgroup rendszerből fejlesztették ki.

1993-ban — a helyzet tisztázása érdekében — a workflow szoftverek szállítói és felhasználói létrehozták a Workflow Management Coalition (WfMC) nevű nemzetközi, nonprofit szervezetet azzal a céllal, hogy szabványosítsa a workflow terminológiát és API funkcióhívásokon keresztül lehetővé tegye a különböző workflow rendszerek együttműködését. Jelenleg az ő ajánlásuk tekinthető a workflow rendszereket leíró szabványnak.

A workflow szegmensben piacvezető CSE Systems a WfMC ajánlásainak figyelembevételével építette fel saját CSE/WorkFlow nevű rendszerét. Több hasonló rendszerrel ellentétben az általa modellezett folyamatok nincsenek mereven beleépítve, s ebben rejlik igazi dinamizmusa is. Bármilyen folyamat programozás nélkül modellezhető vele, így nemcsak egy konkrét területre adaptálható, hanem tetszőleges irodai folyamat algoritmizálható vele.

Teljes körű folyamatkezelés

A WorkFlow felépítésének első és legfontosabb lépése az intézményben folyó irodai folyamatok felmérése és elemi folyamatokra bontása. Itt rögtön meg kell jegyezni, hogy ez a tevékenység nem azonos a munkafolyamatok át-, illetve újraszervezésével. A WorkFlow általános jellegének köszönhetően ugyan neveket rendelünk az elemi folyamatokhoz, ám ezek a nevek csak számunkra hordoznak információt, a rendszer szempontjából tetszőleges stringek. A rendszer ezektől függetlenül építi fel a munkafolyamathoz tartozó struktúrát a felelős személyekkel, osztályokkal, triggerekkel. (A triggerek a rendszer által automatikusan indított programok lehetnek.)

A valóságban a munkafolyamatok nincsenek minden esetben kapcsolatban dokumentumokkal. Egy brókerházban például egy munkafolyamat felépülhet pusztán telefonbeszélgetésekből is anélkül, hogy közben egyetlen papíralapú vagy elektronikus dokumentum keletkezne. A cég állítása szerint a WorkFlow képes az ilyen munkafolyamatok kezelésére is, noha valójában nehezen képzelhető el olyan munkafolyamat, amely közben ne születne legalább egy (pl. a munka elvégzését igazoló)

IQForm bizonylat-feldolgozó

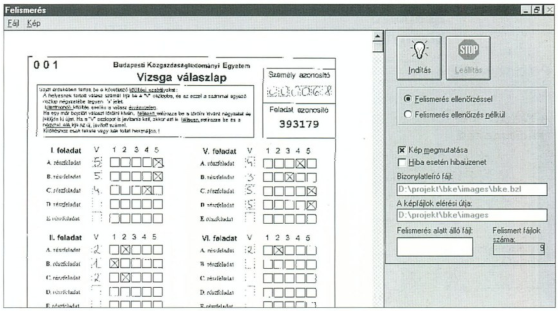
Papírhegyek helyett

A '80-as évek óta visszatérő szlogen a számítástechnikában a papírmentes iroda. Az akkoriban még inkább csak lehetőségként megfogalmazott célkitűzés mára már realitássá vált. A mindenfelől papíron áradó dokumentumokat célprogramok teszik binárisan feldolgozhatóvá, ezek sorába tartozik az IQSoft által fejlesztett IQForm programcsomag is.

Szinte természetes, hogy készpénzbefizetéseinkkel, ajánlott küldeményeinkkel a legközelebbi postahivatalt keressük fel. Az egyes postahivatalok forgalmát naponta összesítik, és ezt megküldik a központi elszámolónak. A belső elszámolási bizonylatok napi kezelése (a pontosság itt a hatalmas forgalom mellett is elemi követelmény!) és az ehhez kapcsolódó statisztikák naprakész vezetése szinte megkövetelte a műveletek számítógépes támogatását. A megoldást az IQSoft által kifejlesztett IQForm programcsomag jelentette. Az IQForm — mint látni fogjuk — rugalmasan adaptálható minden olyan munkafolyamathoz, ahol nagy tömegű papírbizonylaton érkező inputot kell számítógépes rendszer számára is hozzáférhetővé tenni, például bankok, biztosítók, adóhivatalok, társadalombiztosítás, Statisztikai Hivatal stb. Talán nem túlzás azt állítani, hogy széles körű elterjedése javítana komfortérzetünkön, hiszen csökkenne a sorban állás, várakozás a hivatalokban, illetve alkalmazásával javulhatna az egyes cégek hatékonysága.

A Magyar Posta Rt. Napi Elszámolási Rendszere

A Hypermédia Systems Kft.-vel és az IBM-mel közös vállalkozásban megvalósított rendszerben az IQSoft egyik vezető munkatársa, *Kapusy Attila* segített eligazodni. A programrendszer egyetlen MS-Windows 3.x-es PC-ről felügyelhető. Bemenetét az egyes postaigazgatóságoktól a Napi Elszámolási Rendszert (NER) működtető csoporthoz érkező bizonylatok jelentik. Egy nagy teljesítményű Kodak szkennerrel lehet beolvasni a bizonylatokat a számítógépes rendszerbe. Az SCSI illesztésű ImageScan 990-es segítségével



Vizsgalap a BKE-n — feldolgozását IQFormmal végzik

dokumentum. A teljes körű dokumentumkezeléshez hozzátartozik a papíralapú dokumentumok felhasználása is. Az IQSoft megoldotta az általa korábban kifejlesztett DOKTÁR-ArchiWare és a WorkFlow közötti átjárhatóságot, így egy ügy lezárásakor archiválni lehet a WorkFlow belső dokumentumait, és fordítva, egy eredetileg papíralapú, digitalizált dokumentumot ilyen módon be tudnak vonni a munkafolyamatba.

Bármilyen program integrálható

A WorkFlow-ban tárolt dokumentumok elemeihez bármilyen program hozzárendelhető. Ennek alapján az objektumok elemei archiváló, térinformatikai vagy CAD-rendszerekből és más alkalmazásokból származó adatok lehetnek, amiket a felhasználók adminisztrálhatnak anélkül, hogy ehhez szükséges volna a fenti rendszerek működésének ismerete. (A szokásos office- és levelezőprogramokkal való integráció szinte természetes.)

A WorkFlow transzparens módon integrálja a felhasználói objektumba a földrajzilag egymástól távoli szerveren elhelyezkedő objektumelemeket is. Triviális esetben egy szövegfeldolgozást igénylő elem ugyanazzal a programmal állítható elő, nézhető meg, módosítható és nyomtatható ki. Egy képfeldolgozás esetén azonban már több különböző programot kell futtatni. Ennek érdekében a WorkFlow az elemeket kezelő metódusok különböző programjainak definiálását is támogatja.

A definiálást követően a rendszer a megfelelő programokat a megfelelő pillanatban a megfelelő módon fogja használni. Például egy papírdokumentum beolvasásánál a szkenner aktivizálódik, és a képbevitel nem igényel felhasználói beavatkozást, majd elindulhat az optikai karakterfelismerő program is, s az új elem már meg is nézhető, módosítható. A megfelelő alrendszerek integrációjához a forráskód módosítása vagy programozási ismeretek nem szükségesek.

A munkafolyamat-kezelő rendszerekben a dokumentumokhoz tartozó jogosultságok függenek az adott munkafolyamat jogosultságától. Az ilyen elérési jogok az irattartók feldolgozottsági állapota szerint változnak. Egyes dokumentumok a jóváhagyási szintig írható-olvasható állapotban maradnak, de ez után elérhetetlen státusba kerülnek azok számára, akik korábban használhatták őket.

A rendszer kliens/szerver felépítésű, objektumorientált elemeket tartalmaz. A kliensprogramok tipikusan az elterjedt PC-s platformokon futnak, a szervermodulok az ismertebb Unixokon és MS Windows NT-n (3.51, 4.0) Oracle, SQL Server, Informix, Ingres, valamint Sybase adatbázis-kezelővel használhatók.

Dr. Tihanyi Péter, az IQSoft termékmenedzsere arról tájékoztatott, hogy jelenleg két helyen fut a CSE/WorkFlow bevezetési projektje. Az Állami Értékpapír Felügyeletnél egy „telephelyen", a felügyelet irodáiban kerül bevezetésre, míg a MOL Rt. Beruházási Igazgatóságán több helyszínen — Százhalombatta és Szolnok központokkal —, jó néhány telephelyet egyetlen hálózatba integrálva fogják használni a rendszert.

SZABÓ TIBOR

óránként 7000 A4-es bizonylat beolvasása oldható meg, ami a maga nemében egyedülálló teljesítmény. A szkenner automatikus lapadagolóval van ellátva, piros színkiejtéssel dolgozik, és szükség esetén mikrofilmen is tárolja a bizonylat képét. Ez utóbbi tulajdonságát a csekkfeldolgozó rendszernél jelenleg nem használja a posta, minthogy azonban a hatályos joggyakorlat csak a mikrofilmet fogadja el dokumentumként az archiváló médiák közül, más munkahelyeken történő alkalmazás esetén ennek is lehet jelentősége.

A szintén MS-Windows 3.x-en futó beolvasóprogram a beszkennelt bizonylatképek paraméterezés szerinti sokaságát fogja össze és tárolja egy fájlban. A beolvasás során észleli a hátoldalára fordult vagy az elfordult bizonylatokat. Ilyen esetekben a kezelőnek lehetősége van a program által hibásnak tartott kép megtartására vagy újraszkennelésére, megfordítására, s mindemellett a képernyőn egy változtatható méretű ablakban folyamatosan figyelemmel kísérhető a beolvasás.

A bizonylatok képe egy Oracle adatbázisba kerül a képet leíró egyéb paraméterekkel együtt. Ezután az OCR/ICR (nyomtatott/írott karaktereket értelmező) program olvassa be a képeket, és megpróbálja felismerni az adott helyeken kézzel vagy géppel írott számokat. A karakterfelismerő program az adatbázis-kiszolgáló diszkjéről a keletkezés sorrendjében emeli le a fájlokat, s a felismerés eredményét egy leírófájlban helyezi el. A fel nem ismert karakterek helyét csillaggal jelzi. A felismerő program a képeken csak bizonyos képrészeket vizsgál, azokat, amelyeket számára egy leírófájlban előírtak. Ez egy egyszerű szövegfájl, amely a számmezők helyét, a benne található szám minőségét (OCR/ICR) és hosszát tartalmazza. A leírófájl tartalmának megváltoztatásával a rendszer természetesen új vagy megváltozott igényekhez is illeszthető.

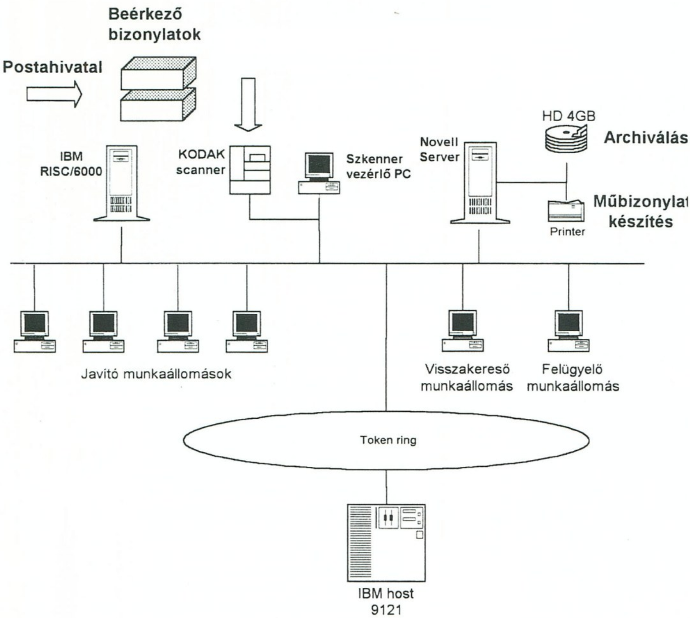
A felismerés sebessége arányban áll a rendszer átlagos teljesítőképességével. Jelenleg egy nyolc OCR és negyven ICR jegyet tartalmazó bizonylat felismerési ideje kevesebb mint fél másodperc. A felismerés megbízhatósága OCR karaktereknél meghaladja a 99,9%-ot, ICR számoknál pedig a 97%-ot.

A rendszeren átmenő bizonylatok adatai a beolvasás és a felismerés után tehát az Oracle relációs adatbázisba kerülnek. Az itt tárolt adatok alapján történik a hibásnak ítélt bizonylatok kikeresése, illetve az egy adott postahivatalhoz tartozó bizonylatok összegyűjtése. Az adatbázisból az adatok a javító munkahelyekre jutnak, ahol a kezelők az esetleges hibákat kijavítják. Innen — egy ellenőrző rendszeren keresztül — a pénzügyi elszámoló központi mainframe-re „utaznak". Az eredménykészítéssel egyidejűleg az adatok és képek beíródnak egy archív adatbázisba, ahonnan később megtekintés céljából visszakereshetők. Statisztikákat a rendszer automatikusan készít naponta, illetve a feldolgozási szakaszok végén, de bármikor kérésre is előállíthatók a kívánt összetételben.

Természetesen lehetőség van az adatok archiválására is; időről időre CD-re írják az Oracle adatbázisban tárolt adatokat. Ezt a feladatot egy fizikailag különálló (PC) szerver kapta, amely gondoskodik az archivált adatok kezeléséről is. Szükség esetén az összes bizonylat képe visszakereshető a vonatkozó adatokkal együtt. A keresés az archívumban tetszőleges mezőkre és tetszőleges feltételekkel elvégezhető.

Oracle adatbázisszerverként egy IBM RS/6000-es kiszolgáló működik, de ez a gép egyidejűleg több munkafázist is teljesít. Az egyes feldolgozási műveletek — úgymint beolvasás, felismerés, adatbázis feltöltése, javítás és eredménykészítés — egymástól időben elkülöníthetők, azaz nagyjából párhuzamosan végezhetők, ami nagyfokú rugalmasságot teremt a napi munka megszervezéséhez.

A postai bizonylat-feldolgozó rendszer hálózata



Merre tovább?

A rendszer remekül bevált a gyakorlatban. Ezt az is fényesen bizonyítja, hogy újabb feladat elvégzését bízták rá, nevezetesen a több bank által is kibocsátott kincstári takarékjegyek forgalmának feldolgozását. (A kincstári takarékjegyek — a kibocsátó bank által meghatározott — különböző címletű megtakarításokat tesznek lehetővé 10 ezertől 1 millió forintig, és nevüknek megfelelően a postahivatalokban válthatók.) A rendszer robusztusságát jellemzi, hogy a többletterhelés mellett is egy műszak alatt végez a megnövekedett teendőkkel. Ehhez mindössze a hibajavító munkaállomások számát kellett tizenhatról huszonnégyre emelni, illetve a Cardinal Kft.-vel közösen kifejlesztett, felismerést végző program az AIX-ről egy Linuxot futtató PC-re került át, míg a többi fázis továbbra is az AIX-en fut. A felismerés jelenleg csak a számokra korlátozódik (kézzel írott és nyomtatott változatok vegyesen), de a két cég már a közeljövő feladatai közé sorolta a nagybetűk értelmezését is (OCR/ICR egyaránt). Ez azért is ambiciózus törekvés, mert a piacon jelenleg elérhető karakterfelismerő programok jó minőségben csak a különböző nyomtatott szövegeket képesek értelmezni, a kézzel írottakat még nem elég jó hatásfokkal.

Szabó Tibor

Beszélgetés Horváth Jánossal, a Miniszterelnöki Hivatal helyettes államtitkárával

A közigazgatás informatikai reformja

A nemrég megalakult Informatikai és Távközlési Kormány Bizottság (ITKB) titkári teendőit a Miniszterelnöki Hivatal új helyettes államtitkára, *Horváth János* látja el, aki egyúttal az Informatikai Tárcaközi Bizottság (ITB) vezetője is. A közigazgatási informatika első számú szakemberét új feladatköréről, a közigazgatásban végbemenő informatikai reformfolyamat lépéseiről kérdeztük.

Hogyan alakul az informatika kezelése a kormányzatban, közigazgatásban, mi változik az ITKB létrejöttével?

H. J.: A rendszerváltáskor az informatika kikerült valamennyi tárca feladatköréből, mégpedig azzal a jelszóval, hogy a piac fogja megoldani ennek szabályozását. A következő egy-két év bebizonyította, hogy a kormányzati szférában ennek a feladatnak a piac nem tud megfelelni. Ekkor alakult meg az ITB, amely a Miniszterelnöki Hivatal irányításával működik, tagjai a minisztériumok informatikai vezetői és egyes olyan szervezetek (Parlament, Számvevőszék, Ügyészség stb.) meghívott vezetői, amelyek nem tartoznak a kormány irányítása alá. Az ITB koordinációs mechanizmusokat alakít ki, közvetett koordinációs eszközöket érvényesít. Hasonlóan az Európai Unióban működő gyakorlathoz, ajánlásokkal, szabványokkal, útmutatókkal igyekszik befolyásolni a fejlesztéseket. Az ITKB megszületéséig hiányzott a kormányzatban az informatikai funkció. A *Lotz Károly, Kiss Elemér* társelnökök vezette új bizottság létrejöttével ez a funkció kormányzati szintre emelkedett. A névben található „távközlési" kifejezés magyarázata: az informatika és a távközlés sok ponton közeledik egymáshoz, sőt áthatja egyik a másikat. A bizottság arra törekszik, hogy a közös felületet gondozza, és ilyen értelemben segítse elő az információs társadalom megvalósulását. Csupán közbevetve jegyzem meg, egyre világosabban látszódnak azok a kontúrok is, amelyek csak az informatikára, illetve csak a távközlésre jellemzőek. Visszatérve: mivel a tevékenység rangja emelkedett, ennek ellátására a Miniszterelnöki Hivatal létrehozott egy kormánybizottsági titkárságot, amelyet helyettes államtitkári minőségben vezetek. Azzal bővült ki a funkció, hogy az informatikai és távközlési termékek központosított közbeszerzésének irányítását, felügyeletét látjuk el a költségvetésből gazdálkodó szervezetek számára.

A kormánybizottság működése révén a kormánynak rálátása van az informatikai és távközlési beruházásokra, a szintén általam vezetett ITB a központi államigazgatásnak, a minisztériumok belső számítógépesítésének ügyeivel foglalkozik.



Milyen közvetlen feladatokat kell irányítania új minőségében?

H. J.: 1997. április elsejétől az informatikai és távközlési termékek tömeges beszerzése központosított közbeszerzés formájában történik a költségvetési szervek számára. A kormány által jóváhagyott normatívákból választhatnak a felhasználók. A központosított beszerzések irányítási feladatait látjuk el, pályázatokat, tendereket készítünk elő, futtatunk le. Ugyanakkor az is látszik, hogy jelentősen változtatni kell a tenderezés gyakorlatán, véget kell vetni annak, hogy általában legyenek megfogalmazva az informatikai eszközbeszerzésekkel kapcsolatos dokumentumok. Ehelyett arra törekszünk, hogy konkrétan, paraméterezetten legyen minden igény meghatározva a tenderekben. Célunk bevezetni a nyugat-európai országokban alkalmazott, EU-konform közbeszerzésekhez hasonló gyakorlatot az informatikai termékekre, mind műszaki, mind szervezési, mind gazdasági és jogi szempontból. Úgy látjuk, a közigazgatási informatikában is kell egy úgynevezett rendszerépítési KRESZ. Minél előbb átvesszük az EU gyakor-

latát, annál hamarabb tudjuk azt alkalmazni.

Hogyan jellemezné a közigazgatásban az informatikai beruházások helyzetét?

H. J.: A beruházások sohasem egyenletesen elosztva jelentkeznek. Az egyik szervezet előreszalad, a másiknál ugyanakkor nem történik meg ugyanabban az időben hasonló célú beruházás. Most éppen azon fáradozunk, hogyan lehet megtalálni azt a kedvező fejlesztési ütemet, amely mindenütt tartható. Néhány tárcánál azt látni, hogy a számítástechnikai beruházások közgazdasági vonatkozásait nem veszik tekintetbe. Ugyanis figyelemmel kell lenni arra, hogy a közigazgatásban háromévenként nem lehet kidobni a rendszereket, újakat beszerezni, és átképezni a szakembereket. Célunk megtalálni azt a formát, hogy a rendszerfejlesztéseket a megfelelő ütemben lehessen biztosítani. Nem igaz az az adat, miszerint a közigazgatás évente több mint 30 milliárd forintot költ informatikára, illetve hogy az informatikai fejlesztés e téren visszaesett. Az idén is, bár lassú ütemben, de tovább nőnek a szervezetek informatikai beruházásai.

Mit terveznek bevezetni az EU-konform közigazgatási informatika megvalósítása érdekében?

H. J.: Amire leginkább koncentrálunk, és mindenképpen végre akarjuk hajtani: szeretnénk bevezetni az Európai Unióban már alkalmazott ún. számítógépkezelői jogosítványt. Ez közigazgatásunkban távlatilag kötelezően megszerzendő lesz, és az autóvezetői jogosítványhoz hasonlóan mindenki, aki megkapja, európai szinten, az EU területén mindenütt használni tudja majd. Néhány év alatt mintegy 50 ezer, a közigazgatásban tevékenykedő ember beiskolázására gondolunk. Most kezdtük el az NJSzT-vel honosítani a számítógépkezelői jogosítvány megszerzésével kapcsolatos EU-dokumentumot. Itt az érdekel bennünket a legjobban, hogy milyen módon lehet elfogadtatni ezt a jogosítványt közigazgatásunkban, hogyan illesszük szabályba a folyamatot.

Milyen feltételekkel, ütemezéssel valósulhat meg ilyen sok ember beiskolázása, vizsgáztatása?

H. J.: Ki kell alakítani a minősítési rendszert, valamint kijelölni az oktató és vizsgáztató szervezetet. Ez is jelenlegi feladataink közé tartozik. Bár az oktatást a piacra bízzuk, a vizsgáztatás feltételei egységes alapúak lesznek. A jogosítvány bevezetésében a fokozatosságot tartjuk szem előtt. Valószínűleg az első időszakban a vizsga ajánlott, a második etapban erősebben ajánlott lesz, és azt mondjuk, hogy 45 év alatt mindenkinek meg kell szereznie a jogosítványt. A finanszírozás és egyéb funkciók szempontjából azt szeretnénk, ha ez a beiskolázási folyamat úgy működne, mint ahogy az a nyelv- vagy autóvezetési vizsgán megszokott. Fontos, hogy közmegegyezés szülessen abban, milyen tartalma legyen a tanfolyamoknak. Ezt az EU-val is egyeztetni kívánjuk.

Mikor kezdődhet és meddig tarthat ez a folyamat?

H. J.: Úgy gondolom, e folyamatnak soha nem lesz vége, hiszen mindig újabb munkatársak kerülnek be a közigazgatásba, akiket be kell iskoláznunk. Az EU-dokumentum honosítását, mint már említettem, megkezdtük, jelenlegi elképzelésünk szerint 1998 végéig mintegy 5-10 ezer embert tudnánk képeztetni. Ebben a munkában mi csak a közvetítő, irányító szerepet szánjuk magunknak, a konkrét oktatást az NJSzT keretében képzeljük el. Egyelőre annyit, hogy a sikeres indulást követően folyamatosan kívánjuk az említett képesítéshez juttatni az összes közigazgatási dolgozót.

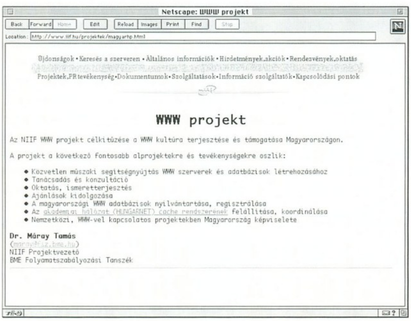
Kovács Attila

A HUNGARNET WWW cache rendszere

Lassú! — hányszor kiáltunk fel bosszankodva, amikor WWW böngészőnkkel egy-egy távoli szerveren kalandozunk, és alig győzzük kivárni, amíg a kért oldal megjelenik a képernyőnkön. Persze hogy lassú, miért is lenne gyors, amikor — szerencsére — olyan sokan, egyre többen, használják a hálózatot, de sajnos a távközlési vonalak kapacitása nagyon korlátozott. A kedvenc WWW pedig sokat fogyaszt, egy-egy szép (vagy csúnya) színes oldal letöltése igen nagy mennyiségű adat átvitelét igényli.

Hogyan lehet segíteni ezen a problémán? Hogyan lehet a „nagy fogyasztású" alkalmazásokat úgy használni, hogy ne terheljük agyon a hálózatot? Természetesen kézenfekvő megoldás lenne a távközlési vonalak kapacitásának megfelelő bővítése. Csakhogy ez elsősorban pénzkérdés, mivel az ilyen vonalak ma Magyarországon hihetetlenül drágák. Másik megoldás a hatékonyság növelése, és ezzel együtt a takarékoskodás. Miután a legnagyobb fogyasztó a WWW, a legjobb eredményt a WWW-forgalom hatékonyságának javításával érhetjük el. Ennek egyik kitűnő módja a WWW cache-ek alkalmazása. Világszerte előtérbe kerültek az ilyen rendszerek, még azokban az országokban is, ahol lényegesen nagyobb sávszélességű Internet kapcsolatok szolgálják a felhasználókat. Magyarországon sincs semmi akadálya annak, hogy minél többen igénybe vegyék ezt a korszerű eszközt, és ezáltal még hatékonyabban használhassák a hálózatot.

Bővebb információt a Weben is olvashatunk a témáról



Hogyan működik a WWW cache?

A HTTP cache szerver használata esetén a WWW kliens (böngésző) nem közvetlenül az eredeti forráshoz fordul, amikor egy objektumot le akar tölteni, hanem egy ún. cache szerverhez (HTTP cache, proxy szerver). A cache szerver ellenőrzi, hogy a kívánt objektum megtalálható-e a saját lokális tárolójában (a cache-ben), és ha igen, akkor azt innen adja és nem a forrás WWW szerverről tölti le. Ezáltal nem kell a sokszor igen leterhelt és ezért lassú működésű nemzetközi vagy belföldi nagy távolságú összeköttetéseket igénybe venni és tovább terhelni. Mivel egy cache szervert sok felhasználó alkalmaz, ezért jó esélye van annak, hogy a kívánt objektumot nem sokkal előttünk valaki már lekérte, így az bekerült a cache-be és ott megtalálható. Egyetlen független cache szerver használata esetén is a találati arány általában >20%, azaz legalább minden ötödik lekérdezni kívánt objektum a cache-ben fellelhető. Ha az objektum még sincs a cache-ben, akkor a cache szerver vagy más közeli cache szerverektől próbálja meg azt letölteni (hierarchi-

Előnyök és hátrányok

Minden éremnek két oldala van, így a WWW cache-elésnek is, de az egyik szerencsére sokkal fényesebb. Vizsgáljuk meg tehát a cache használatából származó előnyöket és hátrányokat is.

Előnyök

• *A WWW dokumentumok elérése gyorsabbá válik.* Mivel a statisztika szerint legalább minden ötödik letöltendő dokumentum a cache-ben már megtalálható, ezek – ha az eredeti forrás nagy távolságra van, vidéki vagy nemzetközi vonalakon érhető csak el – sokkal gyorsabban érkeznek. A többi objektum esetében a letöltés sebessége ugyanakkora, mint cache használata nélkül. (A cache szerverek által hozzáadott késleltetés olyan kicsi, hogy nem érzékelhető.)

• A belföldi és nemzetközi *Internet vonalak túlterheltségének csökkenése.* Nagyon sok dokumentum ugyanis ahelyett, hogy sokszor – minden hozzáféréskor – áthaladna ezeken a vonalakon, csak egyszer okoz forgalmat. A vonalak terheltségének csökkenése azt is jelenti, hogy minden egyéb forgalom – az is, amelyik nem cache-elhető – gyorsulni fog!

• *Költségmegtakarítás.* Az információtovábbítás – különösen nagy távolságra – igen költséges dolog. A megtakarított forgalommal tehát jelentős költségmegtakarítást is el lehet érni.

Hátrányok

• Ritkán ugyan, de megtörténhet, hogy egy-egy WWW oldalnak *nem a legfrissebb verzióját* kapjuk, hanem legfeljebb egy nappal régebbi változatát. Ez akkor fordulhat elő, ha az eredeti dokumentumot megváltoztatták azóta, amióta azt a cache tárolja. Mivel a cache-ek általában legfeljebb egy napig tárolják a HTML szövegeket, majd szükség szerint újra frissítik, a napi gyakorisággal változó oldalak esetén okozhat ez leginkább problémát. A WWW adatbázisok dokumentumainak túlnyomó többsége azonban „statikus", azaz nagyságrendekkel ritkábban változik.

A dinamikusan (az oldalra való hivatkozás pillanatában) generálódó dokumentumokkal *nincsen probléma*, mert ezeket a cache-ek amúgy sem tárolják, tehát mindig az eredeti példányt töltjük le!

Ha gyanús, hogy esetleg nem a legfrissebb dokumentumot látjuk, és ki akarjuk kényszeríteni az eredeti letöltését, a böngésző ***„reload"*** funkciójával ezt bármikor megtehetjük.

• Vannak olyan esetek, amikor egy-egy dokumentum letöltése során a böngészővel nem lehet a „reload" funkciót előidézni, mondjuk olyankor, amikor nem HTML dokumentumot töltünk le, hanem valami mást, rendszerint a diszkre vagy egy külső megjelenítő program számára (pl. amikor FTP helyett használjuk a WWW böngészőnket, és olyasféle fájlokra hivatkozunk, hogy: cikk.ps, MS-CSODA.EXE, szoftver.tar.gz stb.). Ilyen esetekben, ha újra kellene tölteni, nem használhatjuk a „reload" funkciót (mert a böngészőkben nem implementálták), és ezért mindig a cache-ből kapjuk az esetleg elavult vagy hibás fájlt. Ilyenkor az a megoldás, hogy az adott fájl letöltése idejére ki kell kapcsolni a böngészőnkben a cache használatát.

kus cache rendszer) és továbbítani a kérdező WWW böngészőprogramnak, vagy az eredeti forráshoz (WWW szerverhez) fordul.

Az együttműködő cache szerverekből nagy hatékonyságú hierarchikus cache rendszer építhető, ahol a találati arány még nagyobb, mint a független cache szervereknél (gyakran 30% fölé emelkedik). A hierarchikus cache rendszerbe kapcsolt cache szerverek együttműködnek egymással: szomszéd vagy szülő-gyerek viszony alapján kommunikálnak. A párbeszédre egy külön erre a célra kifejlesztett Internet protokollt, az ICP-t (Internet Cache Protocol) használják. A felhasználók WWW kliens programjaiban konfigurálható az, hogy melyik cache szervert vegyék igénybe (lásd *Hogyan használjuk?* c. boxunkat). Nemcsak a HTTP, hanem az FTP- és gopher-forgalom is cache-elhető — a legtöbb cache szerver és WWW kliens erre lehetőséget ad. Mindazonáltal a leghatékonyabb működés — épp a forgalom volumene miatt — a HTTP cache-eléssel érhető el.

A Hungarnet cache hierarchiája

A magyar kutatói IP hálózat, az NIIF által üzemeltetett Hungarnet bekapcsolódhat a Hungarnet hierarchikus WWW cache rendszerébe. Ez a cache rendszer jelenleg max. háromszintű. A felső szinten található az NIIF parent cache szerver, a cache.iif.hu (a helka.iif.hu gépen), amelyen pillanatnyilag a Squid 1.1. verziója fut, és 5 Gbájt diszket, valamint 200 Mbájt operatív memóriát használ az objektumok tárolására. A cache.iif.hu szolgálja ki a nagyobb vidéki és budapesti tagintézményekben már működő lokális cache szerverek kéréseit. Egyes nagy sávszélességű összeköttetéssel ellátott budapesti cache szerverek egymást szomszédokként is használják. A cache.iif.hu elegendő erőforrással és teljesítménnyel, valamint jó hálózati kapcsolatokkal rendelkezik ahhoz, hogy betöltse a központi cache szerepét. A fenti struktúra alkalmas arra, hogy mind a külföldi, mind a belföldi vidéki vonalakon jelentős sávszélesség-megtakarítást érjünk el.

Az NIIF részt vesz a TERENA által koordinált CHOICE projektben, melynek célja a cache rendszerek építésének összehangolása és jövőbeli összekapcsolásuk egy nagy európai cache rendszerré. A Hungarnet cache hierarchiáját, az abba kapcsolódó cache szervereket a tagintézmények felhasználói vehetik igénybe, sőt kell hogy igénybe vegyék, mivel ez a saját érdekük és a közös érdek egyaránt.

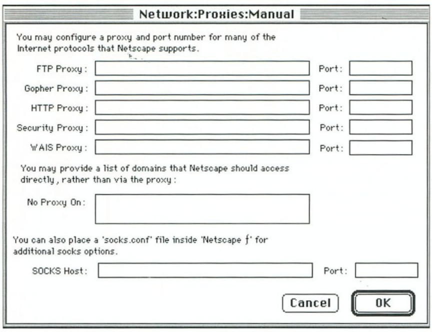
Cache szerverek felállítása elsősorban ott célszerű, ahol a kisebb vagy nagyobb helyi hálózaton (LAN) többen is használják a WWW-t, és a hálózatot a külvilággal (Internettel) összekötő vonal kapacitása időnként vagy rendszeresen szűknek bizonyul — tipikusan tehát intézményekben, cégeknél, Internet-szolgáltatóknál. A cache szervert a hálózat olyan pontján kell felállítani, amely jó (gyors) összeköttetéssel rendelkezik a külvilág felé. A magyarországi akadémiai hálózatot üzemeltető, ill. koordináló NIIF Műszaki Tanácsa erőteljesen szorgalmazza a tagintézményekben a lokális cache szerverek felállítását.

Mi kell egy cache szerver felállításához?

Hardverként egy megbízhatóan üzemelő, Unix operációs rendszerű gép; bármilyen manapság elterjedt architektúra és Unix verzió alkalmas (Linux is!). Nincs szükség túl nagy CPU kapacitásra (sebességre), a lényeg az, hogy legyen elegendő memória (16 Mbájt a cache számára) és diszk (min. 50 Mbájt).

A szükséges diszkkapacitás függ a cache-t használók számától is, de túlságosan túlméretezni nem érdemes a diszkterületet, mert attól *nem* lesz jobb a cache! Nem feltétlenül szükséges a cache szerver számára külön gépet dedikálni. A cache mellett a gépen egyéb alkalmazások is futtathatók.

Hogyan használjuk?



A WWW cache használata igen egyszerű. A legtöbb elterjedt WWW böngésző (kliens) programot (Netscape, Mosaic, Lynx, Internet Explorer) könnyű úgy konfigurálni, hogy cache szervert (proxyt) vegyen igénybe. Figyelem: most nem az adott böngésző lokális cache-éről van szó (amely sokkal kisebb hatékonyságú), hanem az igazi, külső cache szerverről (proxy szerver)!

Az egyetlen adat, amire szükségünk van, a legközelebbi, általunk is használható cache szerver domain neve és a cache szolgáltatás számára felhasznált TCP port száma. (Ez utóbbi leggyakrabban — de facto szabványként — a 3128.) A hálózati értelemben (hálózati összeköttetés szempontjából) vett legközelebbi cache szervernél előfordulhat, hogy földrajzi értelemben nem ez a legközelebbi. Általában egy-egy intézmény, cég üzemeltet cache szervert saját felhasználói kiszolgálására. Rendszerint egy cache szervert csak egy meghatározott, a szerver üzemeltetője által megszabott felhasználói kör vehet igénybe — ennek főként gazdasági okai vannak —, ezért nem minden cache szervert használhat mindenki.

Tételezzük fel, hogy az általunk használni kívánt cache szerver neve: cache.SLD.hu (ez fiktív név, ilyen nevű cache a valóságban nem létezik!). Ebben az esetben a cache használatának konfigurálása a népszerűbb WWW kliensek esetében a következő:

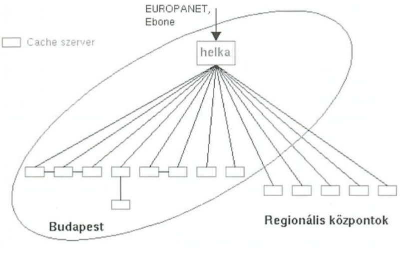
Netscape 2.*, 3.*:
Menüből kiválasztható konfigurációs ablak: Options — Network Preferences — Proxies — Manual Proxy Configuration (View). Az „FTP Proxy, Gopher Proxy és HTTP Proxy" mezőkbe rendre be kell írni, hogy „cache.SLD.hu", a „Port" mezőkbe pedig azt, hogy 3128.

Mosaic 2.7:
Menüből kiválasztható konfigurációs ablak: Preferences — Proxy. A „HTTP Proxy Server, FTP Proxy Server, Gopher Proxy Server" mezőkbe rendre be kell írni, hogy „http://cache. SLD.hu:3128/".

Lynx 2.4:
Állítsuk be a „http_proxy, ftp_proxy és gopher_proxy" környezeti változókat a következő értékre: http://cache. SLD.hu:3128/.

MS Internet Explorer 2.10:
Menüből kiválasztható konfigurációs ablak: View — Options — Proxy.
A „HTTP, Gopher és FTP" ablakokba be kell írni, hogy „cache.SLD.hu", a kis ablakokba pedig azt, hogy 3128.
A legtöbb browser esetében még megadható az is, hogy mit *nem* akarunk a cache-en keresztül lekérdezni (pl. a saját WWW szerverünket).
Ezt követően böngészőnk már használja is a cache-t. Ez valószínűleg meg fog nyilvánulni abban is, hogy sok dokumentum gyorsabban töltődik le, mint korábban. Ha valamilyen okból szeretnénk kikényszeríteni, hogy ne a cache-be már korábban letöltött, hanem biztosan az eredeti forrásból származó legfrissebb dokumentumot kapjuk, akkor használjuk a böngésző „reload" funkcióját. Ekkor a cache újra letölti az eredeti objektumot. A cache szerverek rendszerint max. 24 órán át tárolnak egy objektumot, azt követően — ha az megváltozott — igény esetén újra letöltik. Így semmiképpen sem fordulhat elő az, hogy egy napnál régebbi dokumentumot kapjunk.

A magyar kutatói IP hálózat, az NIIF által üzemeltetett Hungarnet cache hierarchiája



Szoftver tekintetében több elterjedt, akadémiai környezetben ingyen használható cache szoftver van, így például a CERN httpd, az Apache httpd, a Harvest Cached 1.4 és a Squid.

A fentiek közül a CERN httpd alkalmazását nem ajánljuk, mivel teljesítménye lényegesen elmarad a többitől, és hierarchiában is csak korlátozva használható. A Squid 1.1 a legújabb, és a Harvest Cached 1.4 továbbfejlesztéseként jött létre. A Hungarnet hálózaton a Squid cache szerverek üzemeltetését szorgalmazzuk és preferáljuk! (A BME Folyamatszabályozási Tanszék anonymous FTP szervere tükrözi a Harvest Cached 1.4 és a Squid cache szerver szoftvereket, dokumentációkat.)

A cache szerverek felállítása és konfigurálása nem túl nehéz, de feltételezi a Unix operációs rendszer és a TCP/IP hálózat rendszergazda szintű ismeretét. A dokumentációk és a mintakonfigurációk tanulmányozása után a feladat gyorsan elvégezhető. Az esetleg felmerülő kérdések és problémák esetén szívesen állunk rendelkezésre a következő E-mail címeken: *maray@fsz.bme.hu*, *mohacsi@fsz.bme.hu*.

A Hungarnet tagintézményekben üzemelő cache szerver rendszergazdák számára felállítottunk egy levelezési listát is, amelyre az előbbi címeken lehet kérni feliratkozást. A Hungarnet cache hierarchiához való csatlakozás a Hungarnet tagintézményekben üzemelő cache szerverek számára lehetséges. További információ a *maray@fsz.bme.hu* címen kapható.

Tapasztalatok és tervek

A Hungarnet cache hierarchia — fél év kísérletezés és tapasztalatszerzés után — 1996. január 10-től üzemszerűen működik. Igen kedvezőek a tapasztalatok. A hierarchiában több mint húsz szerver üzemel, Budapesten és kilenc vidéki városban (Debrecen, Miskolc, Nyíregyháza, Pécs, Szeged, Veszprém, Gödöllő, Kaposvár, Győr). A hierarchia maximális mélysége három. Az egyes szerverek hatásfoka húsz és negyven százalék között mozog, a terheléstől, időszaktól függően.

A hierarchia csúcsán lévő cache.iif.hu szerver (amely a helka.iif.hu gépen fut) találati aránya szintén eléri ezeket az értékeket. A szerver óránként több ezer cache kérést szolgál ki. Mindennek köszönhetően a Hungarnet vidéki és külföldi vonalain jelentős forgalmat takarítunk meg. A rendszerben mindazonáltal még nagy tartalékok vannak, és ezek kiaknázása úgy lehetséges, ha minél több felhasználó tér át a cache rendszer használatára, és minél több lokális cache-t kapcsolunk a hierarchiába. A Hungarnet cache hierarchia részt vesz a Terena által koordinált Choice projektben, és így — a tervek szerint — a felépülő európai cache hierarchia részévé fog válni.

MÁRAY TAMÁS
MARAY@FSZ.BME.HU

A kihívás: a relációs adatbázis-kezelők kiterjesztése

DB2 — az univerzális adatbázis-kezelő

A relációs adatbázis-kezelő rendszerek képességeinek kiterjesztésében kulcsfontosságú feladat a komplex adatok kezelésének megoldása. Komplex adatok alatt értjük a dokumentumokat, képeket, hang- és videoklipeket, idősorozatokat, térképeket és a felhasználó által definiált adatokat, melyek lehetővé teszik az egyedi üzleti követelmények kielégítését.

Óriási az érdeklődés a *World Wide Web (WWW)* mint alkalmazási platform részéről az olyan adatbázis-kezelők iránt, amelyek ismerik a WWW-s adattípusokat, támogatják a multimédiát, képesek WWW oldalak kezelésére, azok dinamikus létrehozására/frissítésére, illetve a felhasználók WWW oldalakon való böngészésének és egyéb tevékenységének nyomon követésére.

Igényként jelentkezik, hogy az objektummodellezési technikák minél nagyobb támogatottságot kapjanak az adatbázisokban. Ezeknek a technikáknak tartalmazniuk kell az egységbe zárás *(encapsulation)*, többalakúság *(polymorphism)*, öröklődés *(inheritence)* elvét és más objektumorientáltsági elveket, valamint képesnek kell lenniük az objektumorientált alkalmazásfejlesztés előnyeit az adatbázisszerver oldalán is megjelentetni. Ezek a törekvések az objektumorientált és az objektumrelációs adatbázis-kezelők *(OODBMS, ORDBMS)* megjelenésével egyre kiforrottabbá válnak.

Hogyan is néz ki egy objektumrelációs DBMS?

Az objektumrelációs képesség alkalmazás-specifikus jelentéstannal gazdagítja a relációs adatbázis-kezelőket. Ehhez az adatbázis-kezelőnek négy fő területen kell megfelelő kiterjesztéssel rendelkeznie: az adattípusok, a függvények és a metódusok, valamint az indexstruktúrák és a lekérdezés-optimalizálás területén.

Adattípusok

A felhasználó által definiált adattípus *(user-defined type — UDT)* lehetővé teszi, hogy az alkalmazás fejlesztője új adattípust definiáljon. Ez az adattípus alkalmazható az adattábla oszlopaira vagy sorára. Az UDT legalapvetőbb megjelenési formája, amikor az adatbázis-kezelőn belül egy létező adattípust más névvel is megjelölünk. Például az amerikai és a kanadai dollár megkülönböztetésére definiálhatunk egy US_DOLLARS és egy CAN_DOLLARS típust a decimális adattípusok kiterjesztéseként 10 egész és 2 tizedesjegyet tartalmazó méretjellemzővel. Az adatbázis-kezelő megkülönbözteti az általános decimális adattípustól az általunk definiált típusokat. Ezt a képességet az *„erősen típusos"* fogalommal jelölhetjük. Az „erősen típusosság" csak szemantikus összehasonlítást tesz lehetővé két adattípus között. A fenti példát figyelembe véve az US_DOLLARS és a CAN_DOLLARS értékeket tartalmazó oszlopok nem hasonlíthatók össze, bár azonos alaptípusból, megegyező méretekkel lettek származtatva.

A következő lépés az absztrakt adattípusok bevezetése *(abstract data type — ADT)*, amelyek lehetővé teszik tetszőleges összetett belső adatszerkezet és a jellemzők egységbe zárását. Például földrajzi elhelyezkedést meghatározó szélességi és hosszúsági fok jellemzők, vagy a pénzügyi adattípusok mint többdimenziós idősorozatok, melyek egy befektetési portfoliót reprezentálnak.

A sor típus *(row type)* használható a tábla teljes sorának leírására, vagyis ábrázolhatunk általa egyegy entitást (pl. alkalmazott, ügyfél stb.). Jelentőségét növeli, hogy függvény alkalmazható rá, és az adatbázisban a megfelelő sorok hivatkozásai által megállapíthatók az entitások közötti kapcsolatok. A hivatkozási típus *(reference type)* definiálja a kapcsolatot két sor típusú érték között. A hivatkozási típus a teljes adatbázisban egyedi azonosítót rendel a sorokhoz.

Alkalmazásának egyik előnye, hogy a felhasználó a bonyolult kapcsolatok megfogalmazása helyett egyszerű kifejezéseket használhat. Másik előnyének tekinthető, hogy a lekérdezésekben — az elérési út optimalizálása során — a hivatkozásokból származtatott pointerek használhatók alternatívaként a relációs adatbázis-kezelők hagyományos értékalapú kapcsolatai helyett.

A felhasználói adattípusoknak (akár sor, akár oszlop) biztosítaniuk kell az alábbiakat:

- Aggregátum adattípusok — halmazok, listák, tömbök, csomagok — definiálását, mint más típusok gyűjteményét. A gyűjtemények egy speciális típusa a beágyazott tábla — lehetővé teszi, hogy az adattáblának legyen olyan oszlopa, amely egy másik tábla több „sorát" tárolja. Például az „alkalmazottak" táblában tároljuk, hogy egy alkalmazott mely projektekben vesz részt. A „projektek" oszlop minden alkalmazott (sor) esetében tartalmazhatja a projekteket leíró adatok táblázatát.

- Típus-, illetve objektum-hierarchia definiálásának képessége, az altípus/öröklődés támogatásával.
- Adatreplikáció — a DBMS-nek támogatnia kell a felhasználói adattípusokban tárolt adatok replikációját.

Függvények

A felhasználó által definiált függvények *(UDF)* mind az alap adattípusokra, mind pedig a felhasználói adattípusokra alkalmazhatók. Különösen fontosak a függvények az olyan alkalmazások szempontjából, amelyek létrehoznak, kezelnek felhasználói típusú adatokat, vagy hozzáférnek azokhoz. Az UDF-ek támogatják az egységbe zárást, így az alkalmazásoknak nem szükséges ismerniük az adattípusok belső reprezentációját az adatok kezeléséhez.

A felhasználó által definiált függvények típus szerint lehetnek egyszerű függvények, oszlop UDF-ek, amelyek feldolgoznak egy teljes oszlopot és visszatérnek egy aggregátum értékkel (például összeg, átlag stb.), vagy tábla UDF-ek, amelyek tábla jellegű adathalmazt adnak vissza.

A felhasználó által definiált függvények végrehajtásánál megadható, hogy az adatbázis-kezelővel azonos címtartományban működjenek-e, vagy egy elkülönített memóriaterületen. Az előbbi esetén a függvények végrehajtása gyorsabb, míg az utóbbinál a rendszer működése biztonságosabb. A DBMS opciói által szabályozhatjuk az UDF végrehajtásának helyét (kliens, szerver, távoli szerver) és módját (párhuzamos futtatás), illetve speciális jogosultságok is megadhatók a felhasználói függvényeknek az adatbázisszerver részeként történő futtatásához.

A többalakúság támogatása azt jelenti, hogy különböző függvények azonos nevűek lehetnek, a DBMS határozza meg az átadott paraméterek alapján, hogy melyik függvény kerüljön meghívásra. Például a háromszögekhez és a körökhöz is tartozhat egy „területszámító" függvény, bár a területszámítás algoritmusa nem egyezik meg.

Indexstruktúrák

A kiterjesztések harmadik területe a speciális célú indexstruktúrák bevezetése. Az indexek alkalmazásának célja, hogy megnövekedjen az adatelérési sebesség. A hagyományos RDBMS-ek a B-fa indexelési technikát alkalmazzák, melynek alapja egy vagy több oszlop értékeinek összehasonlítása. Például az alkalmazottak fizetésének indexelése lehetővé teszi, hogy kigyűjtsük azon munkatársakat, akiknek a fizetése magasabb évi 600 000 forintnál, anélkül, hogy az alkalmazottak tábla minden sorát megvizsgálnánk. A komplex adatok feldolgozása esetén azonban a B-fa indexelés nem megfelelő. Speciális eljárások szükségesek például a dokumentumokban való keresésre (teljes szövegű indexelés) vagy térbeli adatok esetén (R-fa, rácsozat fájl).

Egy másik fontos jellemzője az indexelésnek, hogy egy függvény kimeneteként is megjelenhet. Ha például egy függvény egy kép színeit határozza meg, akkor az felhasználható index létrehozására is, így a kiválasztási kritérium akár egy szín is lehet.

Lekérdezés-optimalizálás

A lekérdezés-optimalizálás képessé teszi az adatbázis-kezelőt az UDF-ek és az indexstruktúrák hatékony felhasználására. Az RDBMS ismeri, hogy mikor és hogyan használja a B-fa indexelési technikát, illetve hogyan kalkulálja a beépített függvények költségeit. Az új indexstruktúrák megléte és a felhasználó által definiált függvények alkalmazása megköveteli, hogy az optimalizáló modul képes legyen az ezek felhasználásával kapcsolatos költségeket is számítani. Az optimalizálónak — a felhasználó által definiált függvényeken és a keresési metódusokon keresztül — alkalmaznia kell a párhuzamos adatfeldolgozási technikákat, ha azok az adatbázis-kezelő által támogatottak.

Fontos, hogy az adatbázis-kezelő képes legyen létrehozni és használni hivatkozásokat (pointereket) az adatbázis két kapcsolt objektuma között. Így megvalósítható az egyik objektumról a másikra való átlépés a hagyományos értékalapú kapcsolat nélkül. A hivatkozások effektív használata megköveteli, hogy az adatbázis minden sora egyedi azonosítóval rendelkezzen. Ezt az elsődleges és a külső kulcsok önmagukban nem biztosítják.

Nagyméretű objektumok és külső állományok támogatása

Egy objektumrelációs DBMS-nek lehetőséget kell biztosítania a nagyméretű felhasználói objektumok tárolására. Ez történhet az adatbázison belül (BLOB), vagy pedig külső állományokban. Ideális, ha az adatbázis-kezelő a külső adatoknál éppúgy támogatja az azonos integritási szintet, a konzisztens tranzakciót, a mentési és helyreállítási műveleteket, mint a belső adatoknál. Az adatoknak külső fájlban történő elhelyezése több okból is előnyös. Például az adatok „stream"-ként kezelhetők, így a video- és a hangtípusú adatok kliens oldalra való átvitelére a szerver jobban optimalizálható.

Integrált lekérdezések támogatása

Az ORDBMS-eknél egyszerű lekérdezésekkel férhetünk hozzá az adatbázis-kezelő által menedzselt — tetszőleges típusú — adathoz. Szükséges, hogy ezek a lehetőségek egy egyszerű felületen keresztül legyenek végrehajthatók, akár hagyományos, akár komplex típusú adatról van szó. Például egy online vásárlást támogató szoftvernél lekérhető legyen a raktáron lévő összes olyan 42-es méretű férficipő képe, amely túlnyomórészt szürke árnyalatú, mintás, és az ára alacsonyabb, mint 4000 forint.

Nyelvi kiterjesztések

Az ORDBMS-ek számára szükséges, hogy szorosan integrálhatók legyenek az objektumorientált alkalmazásfejlesztő nyelvekkel, méghozzá úgy, hogy megmaradjon a hagyományos 3GL-es és 4GL-es eszközök támogatottsága is. Az integrációnak tartalmaznia kell a kliens gyorsítótárának *(cache)* menedzselését, a létrehozott objektumoknak az adatbázisban történő tárolását és a relációs adatok átadását az alkalmazásnak natív objektumként.

Az adatbázis-forgalmazónak lehetőséget kell teremtenie a DBMS képességeinek kibővítésére, hogy a külső szoftverfejlesztők a lehető legegyszerűbb módon csatolhassák objektumkönyvtáraikat az adatbázis-kezelő környezetéhez. Így a felhasználók egy szélesebb körből választhatják ki az üzleti szempontból számukra legmegfelelőbb megoldásokat.

Az IBM és az objektumrelációs adatbázis-kezelők

Az IBM célja a DB2 szerver kiterjesztése az előző fejezetben említett objektumrelációs tulajdonságokkal. Az objektumrelációs adatbázis-kezelők fejlődését fi-

gyelembe véve a témakörrel kapcsolatos első projektek több mint tíz évvel ezelőtt kezdődtek. Az IBM négy elsődleges területet jelölt meg az objektumrelációs adatbázis-kezelő környezetének kialakításával kapcsolatban *(1. ábra)*.

A DB2 objektumrelációs kiterjesztése

A fejlesztés középpontjában a DB2 szerver állt. A DB2 Common Server Version 2 termék volt az első objektumrelációs képességekkel felruházott IBM adatbázis-kezelő, beleértve a felhasználó által definiált típusok és függvények alkalmazását, a nagyméretű objektumok kezelését, a triggereket és a kibővített integritási megszorításokat. Az IBM lecserélte a teljes lekérdezés-fordító modult és az optimalizálót egy a kiterjesztéseket is kezelő változatra.

Ezeknek az építőelemeknek, illetve a DB2 termékcsaládhoz tartozó DB2 Relational Extenders termékeknek (szöveg, kép, hang, video stb. jellegű kiterjesztéseknek) a továbbfejlesztése és a különböző platformokon (OS/390, OS/400 stb.) történő bevezetése határozza meg a jövő irányvonalait.

Külső állományok csatolása

A külső adatok kezelésében — amelyek fizikailag az adatbázison kívül kerülnek tárolásra — a legfőbb kihívást az adatintegritás biztosítása és a hozzáférés felügyelete okozza. Az IBM-nél erre a célra fejlesztették ki a *„robust file links”* technológiát *(2. ábra)*. Ez a fajta állománycsatolás teszi lehetővé, hogy SQL-en keresztül férjünk hozzá a külső állományokban tárolt adatokhoz. A cél, hogy *integrált lekérdezésekkel* legyenek bővíthetők a már meglévő alkalmazások, és adott legyen az SQL-alapú alkalmazások számára a transzparens hozzáférési mód a külső adatokhoz.

Egy állomány csatolása nem más, mint egy UDT (fájlcsatolás típus), amelynél egy kezelő mutat a külső állományra. A fejlesztő által definiált típussal rendelkező oszlop, létrehozása során, a külső adatokat fogja reprezentálni. A *2. ábrán* látható példában az alkalmazottak fényképei találhatók a külső állományokban, és hivatkozások az alkalmazottak tábla „picture” oszlopában. A DB2 két szoftverkomponens által támogatja a fájlcsatolás típust. Az egyik a „DB2 File-System API”, amelyet a DBMS a külső állományok felügyeletére használ. Például ha egy új alkalmazott kerül „beszúrásra” az adatbázisba, a DBMS ellenőrzi, hogy létezik-e a szükséges külső állomány, és közli az állománykezelő rendszerrel (file system), hogy az ellenőrzött állomány a DB2 adatbázis „tulajdona”. A DBMS ezt a komponenst használja az adatbázis mentésekor a külső állományok kezelésére. A másik komponens a „DB2 File-Link Filter”. Ennek feladata feltartóztatni bizonyos fájlrendszer-hívásokat, hogy biztosítsa a DBMS integritási és biztonsági követelményeit.

Kliens oldali objektumtámogatás

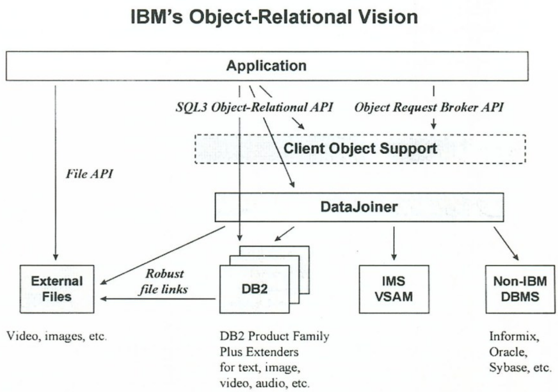
Az IBM objektumrelációs elképzeléseiben több fontos szerepet szánnak a *„Client Object Support” (COS)* komponensnek. Ennek feladata a kliens oldali alkalmazások „objektumos” támogatása. Mindenekelőtt a cél az, hogy lehetővé tegye az adatbázis-kiterjesztések végrehajtását akár a kliens oldalon, akár az alkalmazásszerveren vagy a DBMS-en.

A COS biztosítja minden DB2, fájlcsatolás vagy DataJoiner által hozzáférhető adat logikai nézetét, és garantálja a konzisztens tranzakciókat. A COS menedzseli a kliens gyorsítótárát (cache), a lekérdezések kívánalmai szerint mozgatja az objektumokat az adatbázis és a gyorsítótár között. Intelligenciája lehetővé teszi annak megállapítását, hol optimálisabb a lekérdezések és a felhasználói függvények futtatása (lokálisan vagy a szerveren). Automatikusan pointereket generál az objektumokhoz, így a kliens oldali C, C++ alkalmazásoknak lehetőségük van az objektumon belüli navigációra.

DataJoiner

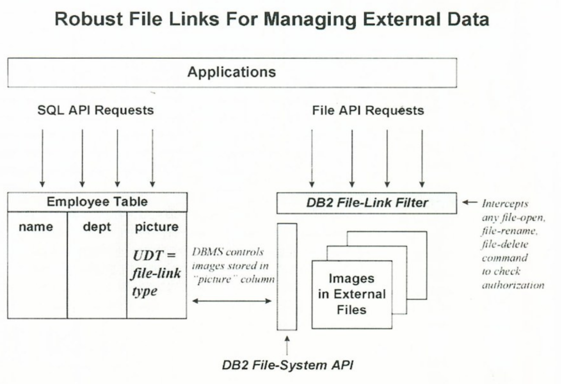
A *DataJoiner* heterogén adathozzáférési megoldást kínáló IBM-termék. Transzparens módon valósít meg írási/olvasási hozzáférést az összes IBM RDBMS termék által kezelt adatbázishoz, továbbá VSAM, IMS, Oracle, Sybase, Informix, Microsoft SQL Server által kezelt adatokhoz, ODBC vagy X/Open CLI felületen keresztül. A DataJoiner nemcsak egyszerű átjáró a DB2 és más adatbázis-kezelők között, hanem egy globális optimalizáló modulon keresztül támogatja is azokat.

A DataJoiner, mivel beépül a DB2 szerverbe, kihasználja annak objektumrelációs kiterjesztéseit, beleértve ezen képességek szimulációját is, más nem IBM adatbázis-kezelőkhöz való csatlakozás esetén.



DB2 univerzális adatbázis-kezelő 1. ábra

Az IBM első objektumrelációs képességekkel is rendelkező adatbázis-kezelője az 1995 júliusában megjelent *DB2 Common Server v2* volt. Ehhez több relációs kiterjesztés is megjelent szöveg, kép, hang- és videoadatok kezelésére. A DB2 Common szerver következő verziója *DB2 Universal Database* néven jelenik meg 1997 első félévében. Ez a v2-es objektumrelációs képességeit ötvözi a *DB2 Parallel Edition* párhuzamos processzálási és skálázhatósági képességeivel, beleértve a szimmetrikus multipro-



2. ábra

cesszálást (SMP), az MPP-szerver (massively parallel processing) alkalmazását és a fürtözési technikát *(3. ábra)*.

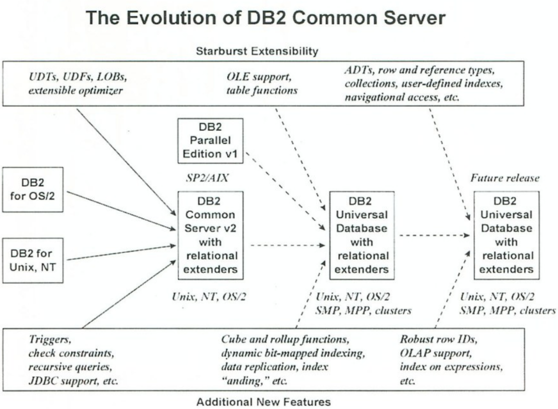
A DB2 Universal Database legfőbb objektumrelációs képességei az absztrakt adattípusok támogatása, a sor típusok, a hivatkozási típusok, a gyűjtemények, a felhasználói indexstruktúrák alkalmazhatóságának biztosítása és az objektumokon belüli navigáció lesznek.

UDT

A DB2 Common Server v2 támogatja a különböző típusokból származtatott felhasználói adattípust, az „erősen típusosságot" és az egységbe zárást. A DB2 relációs kiterjesztései *(lásd cikkünk végén)* támogatják az előre definiált absztrakt szöveg, kép, hang- és video-adattípusokat. A DB2 Universal adatbáziskezelő lehetővé teszi az OLE objektumoknak a DBMS által történő tárolását és kezelését. A jövőben lehetőség nyílik olyan felhasználói absztrakt adattípus definiálására, amely képes a többszörös öröklődést támogatni.

UDF

A DB2 Universalban megjelennek a táblafüggvények, amelyek visszatérési értéke adattábla, és lehetővé válik az UDF-ek párhuzamos végrehajtása. Az UDF-ek bináris formátumban történő „szállítása"



3. ábra

előnyös a külső szofverfejlesztő cégek DB2-alapú fejlesztéseinél. Az UDF-eket futtathatjuk a DB2 szerverrel azonos memóriaterületen, ami sebességnövekedést jelent, vagy a szervertől elhatárolt memóriaterületen, ami biztonságot jelent a még „nem tökéletes" kód futtatásakor.

UDF-eket írhatunk C, Visual Basic, Java, illetve tetszőleges programozási nyelven, amely támogatja a C függvényhívási konvenciót. Támogatott a JDBC API is, így a Java alkalmazások metódusai hozzáférhetnek a relációs adattáblákhoz.

Új indexstruktúrák

A relációs kiterjesztéseken keresztül több új indexstruktúra jelenik meg a szöveges, kép, hang- és videotípusú adatok kezelésére. Lehetőség nyílik az objektumoknak pointereken keresztül való elérésére, az objektumokhoz rendelt attribútumok szerinti indexelésre és a felhasználói indexstruktúrák támogatására.

Optimalizálás kiterjesztése

Az optimalizáló kritikus szerepet játszik a jó teljesítmény nyújtásában. A DB2 optimalizálója szabályalapú, felruházták olyan képességekkel, amelyek lehetővé teszik a lekérdezések finomítását áttranszformálással vagy újraírással.

Csatolt állományok

A DB2 v2 három előre definiált „nagy objektum" (LOB) típust támogat: a bináris (BLOB), a karakteres (CLOB) és a dupla-byte karakteres (DBCLOB) típusokat. Az adatok tárolásával és hiba esetén történő visszaállításával kapcsolatban igen nagy rugalmasságot biztosít a LOB típusok esetén is. A LOB adatok olvasás során közvetlenül az alkalmazás memóriaterületére kerülnek, így csökkenthető a közös memóriaterület túlzott használata. Mindez kiegészül a „robust file link" technológiával, amely garantálja a külső állományok integritását.

Integrált lekérdezések

A DB2 minden objektumrelációs kiterjesztése teljes mértékben támogatott az SQL által, így a felhasználók egyszerű SQL utasításokon keresztül hozzáférhetnek az adatbázis tetszőleges adatához, és képesek a megfelelő függvények felhasználásával keresési műveleteket végrehajtani a komplex adatok körében is.

Támogatja az SQL3 stílusú triggereket, az integritással kapcsolatos megszorításokat és a tárolt eljárásokat. Ez utóbbiak készülhetnek C, Cobol és Fortran nyelveken, 4GL-es eszközzel vagy Javában. Ez a megközelítés lehetővé teszi a programok átvihetőségét, ellentétben a tárolt eljárás programozási nyelvétől. Az IBM tervezi a SQL3 procedurális nyelvi kiterjesztés implementálását, mellyel az eljárások SQL-ben is írhatók lesznek.

A DB2 bővítései

A DB2 objektumrelációs infrastruktúrájába beépülnek a különböző relációs kiterjesztések *(Relational Extenders)*. Ezek mindegyike tartalmaz előre definiált felhasználói típusokat és függvényeket, triggereket, megszorításokat és tárolt eljárásokat. A bővítések által a felhasználó tárolhat szöveges dokumentumokat, képeket, hang- és videoanyagokat a DB2 tábláiban. A tárolt adatok tényleges elhelyezése történhet magában az adattáblában vagy külső állományokban.

Az új adattípusokhoz tartozó attribútumok meghatározzák az adattípus belső struktúráját (pl. dokumentum esetén a „nyelvet", „formátumot" stb.). A bővítések tartalmazzák a típus kezeléséhez szükséges függvényeket (létrehozás, felülírás, törlés, keresés stb.), melyekre a felhasználó SQL utasításokon keresztül hivatkozhat. A DB2 Universal adatbáziskezelőnél mindegyik bővítés beépítésre kerül.

DB2 Text Extender

- Támogatja a teljes szövegű indexelést, a szinonima-keresést. A keresési funkciók vonatkozhatnak szóra, kifejezésre, szomszédságra, tartalmazhatnak helyettesítő karaktereket és más szűkítő vagy bővítő feltételeket.

DB2 Image Extender

- A képek kezelése különböző formátumokban történhet. A keresés során megadhatók különféle színjellemzők és textúrák.

DB2 Audio Extender

- Támogatja a különböző audioformátumú fájltípusokat (pl. WAVE, MIDI), elősegíti az export/import műveleteket, és böngészési lehetőséget biztosít a hanganyagok között.

DB2 Video Extender

- Támogatja a különböző videoformátumú fájltípusokat (MPEG, AVI, QuickTime stb.) és a különböző fájlalapú videoszervereket. Képes a videoklipek lejátszására, és lehetővé teszi, hogy a felhasználó specifikus filmfelvételeket vagy jellegzetes „video-frame"-eket keressen.

KISS ZOLTÁN

Információs rendszer fejlesztése IUS-alapon

Az INFORMIX-Universal Server (IUS) az objektumrelációs adatbázis-kezelők (ORDBMS-ek) családjába tartozik*. A hozzá kapcsolható DataBlade modulok valósítják meg azokat az alap- vagy felépített adattípusokat, amelyeket a kiszolgáló kezelni képes. Ezek a modulok kiszolgáló oldalon merőben új funkcionalitásra és műveletek hatékony teljesítésére adnak módot. Nemcsak a front-end eszközök, hanem az egész DBMS architektúra objektumorientálttá vált.

Cikkünkben először a kiszolgáló, majd a front-end oldali fejlesztési módokat és eszközöket tekintjük át, azután a kapcsolati lehetőségeket felsorolva összefoglaljuk az architektúrát, és bemutatjuk a speciális fejlesztőrendszereket egyesítő objektumorientált CASE eszközt.

Kiszolgáló oldali fejlesztőeszközök

A DB kiszolgálóhoz legszorosabban a DataBlade modulok tartoznak, így természetes, hogy először ezek fejlesztésével foglalkozunk. Webes környezetben az adatbázis-kiszolgálóhoz még különböző funkciójú Web-kiszolgálók is kapcsolódnak az INFORMIX-Universal Web Connect segítségével; ezt a környezetet tárgyaljuk másodikként.

DataBlade modulok fejlesztése

Az objektumrelációs adatbázis-kiszolgálók az RDBMS-ekkel ellentétben nemcsak tárolják, hanem maguk is értik és hatékonyan kezelik a speciális típusú, összetett adatokat. Egy ORDBMS erre olyan adattípusok esetén képes, amilyeneket megvalósító osztálykönyvtárakkal ellátták. Merevebb, az objektumorientált elveket csak korlátozottan megvalósító rendszerekben az elérhető osztálykönytárakat egyszer s mindenkorra beépítik egy-egy adatbázis-kiszolgálóba. Még az sem biztosítható mindig, hogy egyetlen kiszolgáló bármely, a gyártónál elvben elérhető adattípust megvalósítson. Az IUS viszont *objektumorientált értelemben korlátlanul kiterjeszthető*, vagyis a kívánt adattípusokat megvalósító osztálykönyvtárakat — itt DataBlade moduloknak nevezik őket — a felhasználó kapcsolhatja a kiszolgáló magjához, új típusokat is létrehozhat, és egymásra is építheti őket.**

Az elkészített DataBlade modul az adatbázis-kiszolgálóval szorosan összekapcsolható. Általában nincs értelme annak, hogy a DBMS magja és az egyes DataBlade-ek különböző gépeken különböző nyelveken megírva működjenek, fontos viszont hatékony kapcsolatuk; a CORBA-nál jóval speciálisabb és hatékonyabb interfészen keresztül kommunikálnak.

** Az objektumrelációs adatbázis-kezelésről és az Informix objektumrelációs termékeiről lásd az Infopen 1997-ben megjelent számait!*

*** Az IUS-hez az Informix és partnerei által eddig elkészített több mint 30 DataBlade modult az Infopen 1997. márciusi és áprilisi számában ismertettük.*

A DataBlade modulok két nagy csoportra oszthatók. Az *alapmodulok* absztrakt adattípust valósítanak meg: a típusnak megfelelő belső struktúrában tárolják az objektumokat. A *felépített modulok* az alapmodulokra támaszkodva összetett típusokat (pl. struktúrákat és listákat, halmazokat) definiálnak.

Alapmodulok

Az alapmodulok valósítják meg pl. a szabad szöveg, kép, idősorozat, térkép, térbeli mérnöki objektum, mozgókép típusokat. A különböző típusokat megvalósító modulok egymástól nagyon eltérő, speciális tárolási szerkezetet és indexelést alkalmaznak, ami a tömörített reprezentációt és a szükséges műveletek hatékony megvalósítását is lehetővé teszi. Ez a típusspecifikus réteg egy bonyolult, körültekintően meghatározott, általánosan alkalmazható réteget használ fel, amit az adatbázis-kiszolgáló magja valósít meg. Az egyes típusok tárolási mechanizmusai az adatbázis-kiszolgáló magjának a tárgazdálkodására (lemez- és központimemória-kezelés) és tranzakciókezelési (többek között zárolási) mechanizmusára épülnek. Az IUS magja az INFORMIX-OnLine Server megfelelő funkcióit általánosítja és absztrahálja.

A tárgazdálkodási és a tranzakciókezelési funkciókon túl az IUS magja további RDBMS funkciókat is absztrahál, felületeket ad különböző indexelési módszerek — pl. a költségalapú optimalizálás, a felhasználói műveletek, az aggregátumképzés, a helyreállíthatóság — megvalósításához. A DataBlade moduloknak alkalmazkodniuk kell ezekhez a felületekhez. Ha pl. a DataBlade modulban az elérési mechanizmust B-fa indexelésre akarjuk visszavezetni, akkor a modul által kezelt objektumok között definiálnunk kell a rendezési relációval kapcsolatos műveleteket. Mivel ezt többdimenziós világban nem akarjuk biztosítani, ott pl. a kiszolgáló magjába beépített R-fa indexelés követelményeinek (részben rendezéssel, mérettel kapcsolatos műveletek, egyesítés, metszet) kielégítésére törekedhetünk. A kiszolgáló magja nem készülhet föl előre minden elérési mechanizmusra, így az elérési mechanizmusok általánosított felületére is szükség van. Ráadásul összetett objektumoknál gyakran előfordul, hogy valamelyik komponensük vagy számított tulajdonságuk alapján kell megkeresnünk őket, így meg kell engednünk az objektumok attribútumain definiált függvény értéke szerinti indexet is.

Egyes DataBlade moduloknál a megvalósított adattípus felhasználó oldali beviteléhez, lekérdezéséhez, megjelenítéséhez és módosításához kliens API készül.

Bizonyos DataBlade modulokat eleve úgy kell megtervezni, hogy bővíthetőek legyenek. Például képkezelő modul esetében jó, ha egy újabb formátumot a megfelelő konverziós függvényekkel könnyen a meglévők mellé lehet tenni. Természetes nyelvi modulnál a különböző nyelvek támogatását célszerű úgy megvalósítani, hogy az egyes konkrét nyelvek leírásai könnyen csatolhatók, illetve kivehetők legyenek.

Alap DataBlade modulok is felhasználhatják egymást, például képhez tartozhat felirat, amit egy másik modul kezel. Ehhez a képkezelő feliratot elhelyező és előszedő műveletet tesz láthatóvá.

Felépített modulok

A felépített modulok elkészítéséhez más jellegű ismeretekre van szükség, mint az alapmodulokéhoz. Az általánosan felhasználható alapmodulokra támaszkodva speciális célú, összetett típusokat defini-

áló modulok már a DBMS működésével kapcsolatos mély ismeretek nélkül építhetők. Ezt az teszi lehetővé, hogy a felépített modulok nem valósítanak meg új, önálló tárolási és indexelési mechanizmust, így az adatbázis-kiszolgáló magjával csak áttételes a kapcsolatuk az alapmodulokon keresztül.

Az alkalmazások felé a felépített modulok jelentik a hidat. A konkrét alkalmazásokat készítik elő azok a felépített modulok, amelyek egy-egy alkalmazási terület alapproblémáit oldják meg.

DataBlade modulok fejlesztőeszközei

A tulajdonképpeni fejlesztés során a *BladeSmith*-t kell alkalmazni, amivel forrásdefiníciós, deklarációs és make fájlok, valamint csomagoló (a BladePack számára), installáló és regisztráló (a BladeManager számára) scriptek generálhatók. A fejlesztés és a tesztelés nagy része kliensként történik. A DataBlade modulok SQL3, SPL (INFORMIX-Stored Procedure Language), C, C++ és Java nyelven készülhetnek.

Az elkészült modulhoz tartozó fájlokat a *BladePack* eszköz válogatja ki a BladeSmith által kezelt projektek komponenseiből, és elkészíti a terjeszthető diszkettet, CD-ROM-ot vagy tar fájlt.

A *BladeManager* már a felhasználó IUS-hez tartozó eszköze, amellyel a DataBlade modult installálni és regisztrálni lehet a kívánt IUS példányban. A rendelkezésre álló erőforrások függvényében megválasztható, hogy egy DataBlade modult a kiszolgálóval egy gépen (nyilván ez a kedvező eset) vagy más gépen alkalmazzunk (hatékonysági okokból elsősorban felépített modulok jöhetnek szóba).

Az INFORMIX-Universal Web Connect

Az Informix adatbázis-kiszolgálók a Universal Web Connect (IUWC) segítségével kapcsolódhatnak más gyártók Web-kiszolgálóihoz. Az IUWC kiküszöböli a CGI interfészen alapuló megoldások két gyakori hibáját: az állapotmegőrzés hiányát és azt, hogy a Web-alkalmazásnak minden felhasználói hívásnál új példánya töltődik be. Ez rendeli hozzá a felhasználóhoz — annak első megjelenésekor — azt a dinamikus belső azonosítót, ami az adatbázis-kapcsolatot a további párbeszéd során meghatározza, és lehetővé teszi az igényelt tranzakció megvalósítását többlépéses párbeszéd esetén. Maga az IUWC kliens/szerver felépítésű; kiszolgáló része a DBMS-ekhez hasonlóan többszálú, és megosztott memórián működik. Felépítése lehetővé teszi az adatbázis- és a Web-kiszolgáló különböző gépekre helyezését. Az IUWC optimalizálja a kapcsolattartást mind a Web-szerver és a DBMS, mind a Web-kliensek és a DBMS között. A felhasználók számához dinamikusan képes alkalmazkodni., a hozzáférési biztonságot az Internet biztonsági szoftvereivel (pl. a Netscape SuiteSpottal) együttműködve teremti meg.

Az IUWC lehetővé teszi, hogy a Web-hely által szolgáltatott minden információt a DBMS tároljon, és a felhasználói igényeknek megfelelő, pontosan a kért információt tartalmazó HTML lapok generálódjanak dinamikusan.

A Web-lapok dinamikus előállítását HTML-be ágyazott SQL hívások végrehajtásával valósítja meg. Ehhez és egyéb funkciókhoz, pl. IUWC változók vagy hibák kezeléséhez bővíti a szabványos HTML prefixumok körét. Lehetővé teszi, hogy a fejlesztő további prefixeket definiáljon az IUWC alkalmazásban. A HTML prefixumokon alapuló fejlesztést egyszerű környezet, az Application Page Builder segíti. Ha valaki még HTML prefixumokkal sem akar foglalkozni, akkor egy HTML lapot generáló riportfejlesztőt használhat.

A HTML prefixumokon alapuló alkalmazásfejlesztésen kívül a Netscape Server API-hoz és az MS Internet Server API-hoz hasonló programozási környezetet is ad C, C++ és Java nyelvekhez, valamint CORBA IIOP, DCOM osztott környezetekhez.

Front-end fejlesztőeszközök

Az Informix objektumorientált 4GL fejlesztőeszköze a *NewEra*. Rajta kívül az Informix a *Data Director*ral egy sor sikeres terméknek az IUS-hez való natív kapcsolódását támogatja úgy, hogy azok az objektumrelációs kiszolgáló által kezelt új típusú adatokat is könnyen elérjék. Ez a lehetőség nemcsak hatékonyabb az RDBMS-eknél elterjedt ODBC kapcsolatnál, hanem funkcionálisan is sokkal többet kínál, hiszen az csak az egyszerű adatok elérésére ad módot.

Az INFORMIX-NewEra

A NewEra név egyszerre jelent objektumorientált 4GL-t, valamint a nyelvet megvalósító grafikus fejlesztő- és futtatórendszert. A komponensekből való építkezést, az újrafelhasználást segíti, hogy az alkalmazásokba C, C++ vagy NewEra nyelvű osztálykönyvtárak integrálhatók. Az Informix és partnerei kb. tíz osztálykönyvtárat kínálnak.

A fejlesztői és a felhasznált (többek között a beépített) osztályokat, az alkalmazás szerkezetét a Class Browserrel tekinthetjük át. Az alkalmazás komponenseit csoportmunkát támogató repositoryban foghatjuk össze. A NewEra nemcsak multiplatformos, hanem lehetővé teszi particionált alkalmazások kifejlesztését, és így pl. kis erőforrásigényű ügyfél-, alkalmazás-kiszolgáló és adatbázis-kiszolgáló rétegekből álló rendszer elkészítését.

A fejlesztőrendszer (Application Builder és Interactive Debugger része) pszeudokódot használ, míg az elkészült program C++-ra fordul. A fejlesztőrendszer előre elkészített kódrészleteket kínál fel a grafikus eszközökhöz, amelyek nyelvi szerkesztővel finomíthatók/bővíthetők.

A NewEra tartozéka a Seagate Software Crystal Reports jelentéskészítője, és opcionális kiegészítője egy ad hoc lekérdezést támogató eszköz, a ViewPoint. Használatához nincs szükség mély programozói ismeretekre, ezért a NewEra-val ellentétben nincs önálló nyelve.

Az INFORMIX-Data Director család

Az Informix pl. a Visual Basic, a Visual C++, a Visual J++, a Visual Café Pro, a Forte és a PowerBuilder fejlesztők számára teszi lehetővé az IUS natív elérését és adattípusainak használatát. Ilyen együttműködés eredményeként pl. Crystal Reports jelentésbe könnyen (fogd és vidd módon) belehelyezhetünk képet vagy MapInfo beágyazott térképobjektumot. Az alábbiakban a Data Director család Java fejlesztést támogató tagját mutatjuk be.

A Data Director for Java eszközzel bármely javás fejlesztőkörnyezetet (pl. a Symantec Cafét, a Sun Microsystem JavaWorkshopját vagy a Microsoft Visual J++-t) kiegészíthetjük annak érdekében,

hogy (webes környezetben működő) Java programkákat fejlesszünk, amelyek az IUS-t érik el. Grafikus környezetben olyan programkákat generálhatunk, amelyek SQL3 típusú adatok bevitelét, tartalom szerinti kezelését, OLTP-t vagy OLAP-ot tesznek lehetővé.

Objektumorientált, elosztott architektúrájú alkalmazások fejlesztése

Láttuk, hogy az Informix és partnerei által kínált kiszolgáló oldali és front-end eszközök mindegyikéhez specifikus fejlesztőeszközök tartoznak. Ezek azonban csak az egyes komponesek megvalósítására adnak módot. Az egész rendszer kialakításához és a komponensek feladatainak a meghatározásához általános CASE eszközre van szükség, amely a tárgyrendszerek képességeit kihasználandó maga is objektumorientált. A Cayenne (a Westmount, a Cadre és a Bachmann cégek összeolvadásával jött létre) ObjectTeam eszköze a szoftverek teljes életciklusát támogatja, amelybe a tárgyrendszerek fejlesztőeszközei mint lower-CASE eszközök integrálhatók. előállítható, így pl. más eszközzel készült komponensek a projektbe integrálhatók. Iteratív fejlesztés során a konkretizált leírásban végzett változtatások alternatív ágon újra felkínálódnak, nem vesznek el. A konfigurálható ellenőrzési lehetőségek és az egységes repository a fejlesztési technikák összehangolt alkalmazását biztosítják, s mind a repository, mind a kódgenerálás, mind az ellenőrzési lehetőségek dokumentált interfészen keresztül módosíthatók és bővíthetők.

A felhasználó igényei szerint bővíthető menük, űrlapok és diagramok rendelkezésre állnak minden fejlesztési fázisban, hogy a repository új részei és az új funkciók a felhasználói felületen elérhetőek, illetve a generátum finomítását lehetővé tevő lower-CASE fejlesztőeszközök integrálhatók legyenek. Az összetett dokumentumok készítését támogató munkapad, az *Interleaf, FrameMaker* DTP eszközökkel vagy a *WordPerfect*tel, *MS-Word*del, a *DocIt* és a *DocExpress* eszközökkel integrálható DBMS eszközök mellett az ObjectTeam kódgenerátorokkal is rendelkezik C, C++, Smalltalk, Ada, Java, CORBA IDL és Visual Basic nyelvekre. Az Informix adatbázis-kezelőkkel való integráltság mélységét jelzi, hogy mind a back-end SQL-jére (tárolt procedúrákat beleértve), mind a beágyazott SQL-re, NewEra-ra kódot generál módosítható minták alapján, sőt a NewEra fejlesztőrendszerével integrálható.

Többszintű 3GL, 4GL, intra- és Internet alkalmazások fejlesztése

Az információs rendszer egyes komponensei a leggyakoribb esetekben már a beépített hálózati kiegészítőkön keresztül kapcsolódnak egymáshoz. Mit fednek le ezek a beépített lehetőségek?

- A cikkben szereplő front-end eszközökön kívül is sok cég terméke kapcsolódik az Informix kiszolgálók által nyújtott natív vagy ODBC felülethez.
- Az IUS teljes Java támogatást ad. Kliens oldalról kezdve natív Java API-t és JDBC-t is megvalósít. Mint már láttuk, webes környezetben ez a támogatás a Data Director for Javával és az Universal Web Connecttel egészül ki kliens, illetve szerver oldalon, az IUS-t pedig bővíthetjük Javában írt DataBlade modulokkal.
- Az INFORMIX-NewEra alkalmazás particionálást tesz lehetővé.
- Az Informix adatbázis-kiszolgálók a kétfázisú érvényesítésen kívül többféle technikán alapuló, rugalmasan szabályozható replikációra adnak módot: mind a replikálandó rekordok, mind a replikátumok (mely helyeken jönnek létre a másolatok, változtatható-e a replikátum, vagy csak olvasható), mind a replikálás módjai (pl. szinkron vagy aszinkron) kijelölhetők.
- Mint korábban láttuk, az adatbázis-kiszolgálók az INFORMIX-Universal Web Connect segítségével kapcsolódhatnak más gyártók Web-kiszolgálóihoz. Ezenfelül az INFORMIX-OnLine Workgroup Serverhez a Netscape FastTrack Web kiszolgálója és Navigator Gold böngészője is jár, így minden adott egy kísérleti webes adatbázis-alkalmazás kifejlesztéséhez. Ez a megoldás akár intranet, akár Internet irányban továbbfejleszthető hatékonysága, méretezhetősége és pl. biztonsági szolgáltatásokkal való bővíthetősége miatt.

Ahol a termékekbe beépített kapcsolati képességek nem elegendők, ott a következő lehetőségek állnak rendelkezésünkre:

- OSF DCE szabványú Call Level Interface: ODBC + adathozzáférési biztonság/titkosítás (4 fokozat közül választhatunk) + name service funkciók;
- X/Open XA szabványú interfész tranzakciós monitorok felé;
- DRDA interfész IBM mainframe-ek felé;
- EDA/SQL ipari szabvány gateway technológia (több mint 60 adatbázis-kezelő és fájlkezelő rendszer felé).

Fokozatosan egyre nagyobb jelentőségre tesznek szert a Webbel is kapcsolatos ipari szabványok általánosságuk és nyelvfüggetlenségük miatt:

- az XA interfészt és a DCE szabványt is általánosító Object Management Group CORBA 2.0 és webes specializációja, a CORBA IIOP;
- az OMG-nek ellenlábasa a Microsoft; a CORBA-nak itt a DCOM felel meg.

Bár általában a MS ActiveX-et is támogatják a jelenlegi CORBA implementációk, ezek még nem eléggé érettek:

- az elterjedt, speciális middleware-ekkel összehasonlítva erőforrásigényesek, és hatékonyságuk gyengébb;
- a különböző gyártók ORB megvalósításai nem átjárhatók.

A későbbiekben remélhetőleg átjárhatóvá és hatékonyabbá váló különböző ORB megvalósítások fokozatosan beépülnek a middleware-be (lásd pl. az INFORMIX-Universal Web Connect CORBA, DCOM, C, C++ és Java támogatását).

Az ObjectTeam elosztott architektúrájú eszköz, amely felhasználói csoportok hozzáférési jogainak szabályozásával is támogatja a csoportmunkát. A felhasználói követelmények explicite megfogalmazhatók, és a fejlesztés folyamán fokozatos konkretizálásuk/megvalósulásuk követhető. A komponensek újrafelhasználását az objektumorientáltságon túl verzió-, sőt konfigurációkezelés is segíti; konkrét komponesek absztrakciója (pl. programrészek, osztálykönyvtárak szerkezete diagram formájában)

BALOGH KÁLMÁN
KBALOGH@INFORMIX.HU

A Jáva és a HTTP protokoll

Bár az elmúlt hónapokban az elosztott rendszerek programozásának különböző, néha nagyon újszerű, kifinomult módjaival ismerkedhettünk meg, azért nem szabad lenéznünk a — viszonylag egyszerű — HTTP által kínált lehetőségeket. Noha nagyon ügyes, komplex kliens-szervereket írhatunk Jávában, van ezekkel egy kis gond: valahogy telepítenünk kell a kiszolgálónkat egy másik gépre, hogy kapcsolatba léphessünk vele. Viszont az Interneten 10-100 ezer olyan kiszolgáló található, amelyik szívesen fecseg velünk a HTTP protokoll segítségével. Ezek a Web-szerverek. Így nekünk „elég" csak az ügyféloldalt programozni, a szerverek már készen állnak.

Van a *HTTP* protokollnak még egy, az elterjedtségéből következő nagy előnye: manapság az Interneten bolyongva lépten-nyomon tűzfalakba *(firewall)* botlanak a szegény bitek. A tűzfalak meglehetősen finnyás programok, az új, trükkös protokollokat — *IIOP, RMI* vagy akár saját találmányú — keményen kizárják, de arra legalább esélyünk van, hogy a HTTP protokollt, persze megfelelően konfigurálva a rendszert, átengedik. Ezért amíg a tűzfalak gyártói nem készítenek a fenti protokollokhoz ún. köztes *(proxy)* kiszolgálót, a HTTP nyújtja az egyedüli általánosan elterjedt lehetőséget arra, hogy a nagyvilágból (Internet) beláthassunk a belső hálózatok (intranet) féltve őrzött birodalmába.

A HTTP protokollról

A HTTP *(HyperText Transfer Protocol)* köti össze a Webet, ez az a közös nyelv, amelynek segítségével a böngészők és a Web-kiszolgálók szót értenek. A protokollok hierarchiájában a HTTP is a legfelső, az ún. alkalmazási szinten található (mint pl. az *FTP* vagy a *Telnet*), és ez is a megbízható, kapcsolatorientált szolgáltatást szállítani képes *TCP* protokollra épül.

A teljes protokoll ismertetése bőven meghaladná cikkünk kereteit, így kissé önkényesen kiemelek néhány tulajdonságot:

- Jelenleg a HTTP 1.0-s változata terjedt el széles körben, az ennek hibáit javító 1.1-es verzió még csak a fejlesztgetés stádiumában van, elfogadott „szabványa", elterjedt implementációja nincsen.
- A kommunikációt minden esetben a kliens (böngésző) kezdeményezi. Kiépít egy TCP kapcsolatot a szerverrel, ezen átmegy egy HTTP kérés (parancs), amelyre a szerver válaszol, és a válasz végén lebontja a kapcsolatot.
  Ez egyébként a protokoll legnagyobb problémája. A böngésző nemcsak minden egyes lapot, de a lapokon lévő egyéb objektumokat — képeket, hangot, Jáva programkák kódját stb. — külön-külön kéréssel tölt le, és egy TCP kapcsolat kiépítése meglehetősen lassú folyamat. Ezenkívül a fejlesztők nagyon nem szeretik azt a tényt, hogy egy újabb kérésről sehogyan sem lehet megállapítani, vajon egy megelőző „logikus" folytatása, vagy esetleg egy másik felhasználótól érkezik. A HTTP nem hordoz a kérések sorozatát azonosító állapotinformációt. Igaz, hogy kitalálták az ún. „sütiket" *(cookie)*, de ez sem az igazi megoldás.
- Két fontos parancsot használhatunk a protokollban: a *GET*-et, amellyel kéréseket, illetve a *POST*-ot, amellyel az egyszerű kérés mellett kiegészítő információkat — akár egész állományokat — is küldhetünk a szervernek.
- A protokoll-kérések és -válaszok feje szöveges, szószátyár, emberi fogyasztásra készült, de szerencsére az üzenetek törzse „8 bites", azaz a bájtok speciális kódolás nélkül is átmennek.

Végezetül lássunk egy — kissé egyszerűsített — példát egy *GIF* képállomány letöltésének folyamatára. A kliens kérése egyetlen sorból (*GET* parancs, az állomány elérési útja és a protokoll verziószáma) is állhat:

```
GET /images/nude.gif HTTP/1.0
```

A szerver erre a következő sorozattal válaszolhat:

```
HTTP/1.0 200 OK
Date: Thu, 22 May 1997 20:40:00 GMT
Server: Oracle_Web_listener2.1/1.20in2
Content-type: image/gif
Content-Length: 3452
[Itt jön 3542 bájt, a kép maga]
```

Az első sor számmal (200) és betűkkel (OK) közli, hogy sikerült kiszolgálni a kérést. A dátum és a kiszolgáló (Web-szerver) azonosítása után jön a válaszban érkező információ típusa *(image/gif)*, az ún. MIME kódolással megadva. Ezt követi az információ hossza — a fej nélkül — bájtokban, majd maga a bájtsorozat. A rendes kiszolgálók, amennyiben tudják, közlik a bájtsorozat hosszát, de ez nem kötelező, hiszen a kliens abból is észreveheti, hogy véget ért a válasz, hogy a szerver lezárja a kapcsolatot.

HTTP támogató osztályok

A fenti példából látható, hogy a HTTP protokoll TCP-re alapuló implementációja a *java.net.Socket* és *java.net.ServerSocket* osztályok segítségével nem jelentene túlzottan nagy problémát, de a java.net csomag tartalmaz néhány HTTP specifikus osztályt is:

- *java.net.URL*
  Az osztály feladata, hogy egy *URL*-lel *(Universal Resource Locator)* leírt objektumot reprezentáljon. A konstruktoron, ill. az URL cím különböző összetevőinek kezelésén túl három módszert emelnék ki: az *openStream* egy bájtfolyamot nyit meg, amelyből az URL által reprezentált címről érkező válasz bájtjait olvashatjuk, a *getContent*tel a bájtfolyam helyett Jáva objektumként kaphatjuk vissza a választ (l. később).
  A legizgalmasabb talán az *openConnection*, amellyel az URL révén megnyitható kapcsolatot ragadhatjuk meg egy absztrakt *URLConnection* osztályból leszármazott osztály példányának segítségével. Hogy miért érdekes ez? Az URL első része a kapcsolatot leíró protokollt definiálja, ami manapság leggyakrabban a HTTP, pl.

```
http://www.eunet.hu/infopen/cikkek/java/agent.html
```

ám a http helyére más jól ismert, de akár saját protokoll nevét is írhatjuk, mint pl. *ftp, gopher, news.* Bár a Jáva alapkörnyezet csak a HTTP protokoll kezelésére tartalmaz *„connection"* osztályt, a könyvtárak dinamikusan bővíthetők protokollkezelő *(protocol handler)* modulokkal. Sajnos a nagy böngészőgyártók még nem haraptak rá erre a lehetőségre.

- *java.net.URLConnection*
  Ez ugyan egy absztrakt osztály, tehát közvetlenül nem hozhatunk benne létre egyedeket, de a fenti *URL.getConnection* révén az alkalmazandó protokollnak megfelelő kezelőpéldányt kapunk vissza. Ennek segítségével még a szervernek történő elküldés előtt módunk van a kérés különböző paramétereinek beállítására (pl. a POST parancs megadására), és a visszaérkezett válasz feldolgozására is sokkal finomabb lehetőségek adódnak. Többek között nem csupán a válasz bájtokban mért hosszát *(Content-Lenght)* kaphatjuk meg, hanem a típusát *(Content-type)* is. Sőt az itt is meglévő *getContent* módszer használja ezt az információt, és megfelelő típusú, megkonstruált objektumot ad vissza. A lehetséges beépített, automatikusan előállítható objektumok típusa itt sem túl bőséges — jobbára a különböző képformátumok *java.awt. Image* objektumokká történő átalakítása működik —, ám lehetőségünk van a tartalomkezelők *(content handler)* dinamikus bővítésére.

Olvasás Web-kiszolgálóról

A következő rövid példaprogram egy olyan Jáva alkalmazás, amely az első paramétereként megadott URL tartalmát letölti és kiírja a szabványkimenetre. A program nem túl elegáns, csak a statikus *main* fő eljárást tartalmazza:

```
import java.net.*; import java.io.*;
class CopyURL
{ public final static int BUFFER_LENGTH = 4096;
static URL url = null;
static URLConnection con = null;
static BufferedInputStream input = null;
static byte [] buffer = null;
static int fileLength = 0;
public static void main (String[] args){
```

Megnézzük, hogy a programnak van-e egyáltalán paramétere, majd ennek a szövegnek a segítségével létrehozunk egy új URL példányt. A *MalformedURLException* hibás URL szintaxisra utal:

```
if (args.length != 1)
error(„java CopyURL <URLtoLoad>");
try { url = new URL(args[0]); }
catch (MalformedURLException e) {
error(„Wrong URL „ + args[0]); };
```

Következő lépésként megnyitjuk a kapcsolatot, és megragadjuk a hozzá tartozó bemenő bájtfolyamot. Érdemes megfigyelni, hogy az egyszerű *InputStream* objektumunkat egy szűrő osztályon, a *BufferedInputStream*en keresztül használjuk. Igaz, ez az osztály nem ad túlzottan ügyes beolvasó eljárásokat, csak bájt- vagy karaktertömböt lehet belőle olvasni, de a belső pufferelés miatt sokkal gyorsabb, mint például az „intelligensebb" *DataInputStream*:

```
try { con = url.openConnection(); input = new
BufferedInputStream(con.getInputStream()); } catch
(Exception e) { error(e);}
try { buffer = new byte [BUFFER_LENGTH];
```

A másolás nagyon egyszerű: a *read* igyekszik teljesen feltölteni a *buffer*t, és visszaadja, hogy hány bájtot olvasott, a *write* a bufferből ennyit ír majd ki. A ciklus után — ahogy illik — lezárjuk a bemenetet:

```
int lgth;
while ((lgth = input.read(buffer)) != -1)
System.out.write(buffer, 0, lgth);
input.close(); input = null;
} catch (IOException e) {error(e); } }
```

Az általános hibakezelő eljárás kiírja a hibaüzenetet, lezárja a bemenetet, ha korábban nyitva volt, és kilép a programból:

```
static void error (String msg)
{ try {
System.err.println(„CopyURL: „ + msg);
if (input != null) input.close();
} catch (Exception e) {};
System.exit(1); }
static void error (Exception e) {
error(e.toString()); }}
```

Írás Web-kiszolgálóra

Ez az ügyfél oldalán nem sokkal bonyolultabb, de a küldött információ fogadására fel kell készíteni a szervert. Különböző szerverek esetén ez máshogyan történhet, azonban egy trükköt mindegyikük ismer: bizonyos URL-ekre hivatkozva a szerver elindíthat egy — ún. *CGI* — programot, amely fogadhatja a kérést, az átküldött információt, ezt feldolgozza, tárolhatja és válaszolhat az ügyfélnek.

Külön Webre írást bemutató példa helyett nézzünk inkább egy összetettebb példát, ahol az írásnak is fontos szerep jut.

Példaprogram: jegyzetfüzet a Weben

Példánk egy jegyzettömböt igyekszik szimulálni, amit mindenüvé — ahová a Web elér — magunkkal vihetünk. A jegyzettömb fejléccel ellátott lapokból áll, a program segítségével a tömb összes feljegyzésének fejlécét lekérdezhetjük, letölthetünk egyes feljegyzéseket, vagy újakat írhatunk és tárolhatunk a kiszolgálón. A program az egyszerűség kedvéért egyéb feladatokra, pl. feljegyzés módosítására, törlésére nem képes, de remélhetőleg a lényeget így is illusztrálja.

A feladatot egy böngészőben futó programkaként, cifra kezelői felülettel kellene megvalósítani, s bár az *AWT* és *applet* könyvtárak erre minden lehetőséget meg is adnak, az elegáns program eléggé sok munkát igényel, a kód el sem férne a cikkben. Ehelyett csak egy segédosztályt közlök, amely elintézi a kommunikációt, és kezelői felületbe illesztése nem jelenthet túl nagy gondot. Három nyilvános módszert definiáltam a három fenti feladathoz: a *getSubjects* visszaadja a fejlécek listáját, a *getNote* egy adott sorszámú feljegyzést, az *addNote* pedig új feljegyzést tárol.

A szerverről most csak annyit tételezek fel, hogy az összes feladatot egyetlen URL-en található program hajtja végre, ahol a konkrét tevékenység kódját, paramétereit a szokásos HTTP konvenciókkal az URL végéhez csapva adjuk át (a részleteket lásd később):

```
import java.io.*; import java.net.*; import
java.util.*; public class NoteClient
{ String server;
Exception ex = null;
```

A konstruktor csupán elmenti a szerverhez vezető URL-t:

```
NoteClient (String server) { this.server =
server;}
```

A következő módszer egy szövegekből álló dinamikus tömbben *(Vector)* adja vissza az összes feljegyzés fejlécét. *Null*-lal tér vissza módszerünk, ha valami baj történt, de — akárcsak a következő nyilvános módszerek — elnyeli az összes futás közben előálló hibát *(Exception)*. Az utolsó hiba — szükség esetén — a *getLastException* módszerrel visszakapható:

```
public Vector getSubjects()
{ Vector subjects = null;
String lines, nextLine;
ex = null;
```

A szerverről történő olvasást a *getLines* módszer rejti, amely paraméterként nem csupán a szerver címét kapja meg, hanem hozzácsapjuk azt a kiegészítő

paramétert is, amelyik megmondja, hogy most csak a fejlécek érdekelnek.

A CGI programokhoz a paraméterek eljuttatásának leggyakoribb módja (a *GET* módszer) az, hogy az URL végére <név>=<érték> párokat csapunk, az URL-től kérdőjellel, egymástól a „ampersand jellel" (&) elválasztva azokat. Mind a névben, mind a paraméterekben csak egy szűkített kódkészlet fordulhat elő, a térközöket összeadásjellel, az egyéb gyanús karaktereket a %XX karaktersorozattal kódolják, ahol XX a karakter hexadecimális kódja. Szerencsére a példát úgy konstruáltam meg, hogy ilyen konverzióra ne legyen szükség, bár a *java.lang.URLEncoder.encode* módszer ezt megcsinálja.

```
lines = getLines(server + „?cmd=subjects");
if (lines == null) return null;
```

A fenti módszer egyetlen szövegként adja vissza a beolvasott információt. Ezt a következő programrészlet soronként szétdobja egy vektor elemeibe. A *Vector* dinamikus adatszerkezet, méretét nem kell előre megadni. A sorokra bontás egyik lehetséges megoldása, hogy egy *DataInputStream* objektumot használunk, amelynek alapját a beolvasott szövegből képzett *StringBufferInputStream* objektumra építjük:

```
subjects = new Vector(); DataInputStream subs =
new DataInputStream(new
StringBufferInputStream(lines)); try {
while ((nextLine = subs.readLine()) != null)
if (nextLine.length() > 0)
subjects.addElement(nextLine);
} catch (IOException e) {}; return subjects; }
```

Álljon itt példának a lezajlott kommunikáció (a válasz törzsét, a fejléceket kihagytam):

```
GET /cgi-bin/notes?cmd=subjects HTTP/1.0
User-Agent: Java1.0.2
Accept: text/html, image/gif, image/jpeg, *; q=.2,
*/*; q=.2
HTTP/1.0 200 OK
Date: Thu, 22 May 1997 20:40:00 GMT
Allow: GET, POST
Server: Oracle_Web_listener2.1/1.20in2
Content-type: text/plain
Content-Length: 49
[49 bájt hosszú tartalom]
```

Az egyes feljegyzések beolvasásához felhasználjuk az általános eljárást, mindössze a szerver URL-ben adunk meg más funkciót *(cmd=fetch)*, illetve a beolvasandó jegyzet számát (például *ix=2*):

```
public String getNote (int i)
{ ex = null;
return getLines(server + „?cmd=fetch&ix=" +
String.valueOf(i)); }
```

Új feljegyzés tárolására egy másik általános eljárást, a *saveLines*-t használom. A módszer első paramétere szokás szerint a szerver címe, beleértve az *add* parancsot is, a második az elmentendő szöveg, ahova összemásoltam a fejlécet és a jegyzék törzsét:

```
public boolean addNote (String subject, String
note)
{ return saveLines(server + „?cmd=add", subject +
„\n" + note); }
public Exception getLastException () { return ex;}
```

A szövegek beolvasására szolgáló eljárás gyakorlatilag megfelel a fenti *CopyURL* példának:

```
private String getLines(String scriptURL)
{ String next; byte[] buffer = new byte[2000];
int chunkSize; String result = new String();
BufferedInputStream in = null; try {
URL url=new URL(scriptURL);
URLConnection conn = url.openConnection();
```

Az egyedüli igazi újdonság a következő hívás, amely ráveszi az alatta lévő protokollt, hogy közvetlenül a megjelölt szerverről olvasson, ne használjon közbülső átmeneti tárat:

```
conn.setUseCaches(false);
in = new
BufferedInputStream(conn.getInputStream());
while ((chunkSize = in.read(buffer)) != -1 ) {
result = result + new String(buffer, 0, 0,
chunkSize); } } catch (Exception e) { ex = e;
result = null;
} finally { if (in != null) try { in.close(); }
catch (IOException e) {}; return result; } }
```

A szerverre írás általános eljárása:

```
private boolean saveLines(String scriptURL, String
noteText)
{ DataInputStream in = null;
DataOutputStream out = null;
String ack;
try {
URL url=new URL(scriptURL);
URLConnection conn = url.openConnection();
```

A kapcsolat megnyitása után be kell állítanunk, hogy írni is akarunk, azaz POST módszert használunk. Erre szolgál a *setDoOutput(true)*. Az inputra ugyanez „elvileg" felesleges, ez az alapértelmezés:

```
conn.setDoOutput(true); conn.setDoInput(true);
conn.setUseCaches(false);
```

Természetesen először az írást kell megtennünk, hiszen ez kerül az elküldendő kérés fejébe. A kimeneti folyam lezárása utasítja a protokollt, hogy küldheti a fejet. A kiküldött információ hosszát a programban nem kell megadnunk, ezt az eljárások maguktól kiszámítják és beírják a fejbe:

```
out = new
DataOutputStream(conn.getOutputStream());
out.writeBytes(noteText); out.writeBytes(„\n");
out.close(); out = null;
```

A kérés elküldése után olvashatjuk a szerver válaszát. Egyszerű példaprogramunk nem vár válaszra, ezért csak beolvas egy sort, de nem törődik annak tartalmával:

```
in = new DataInputStream(conn.getInputStream());
ack = in.readLine(); in.close(); in = null;
return true; } catch (Exception e) {
ex = e; return false; } finally { try { if (in !=
null) in.close(); if (out != null) out.close();
} catch (IOException e) {} } }
```

Álljon itt is egy kommunikációs példa. A válasz törzsét ennél is lehagytam, de a kérés törzsében megtartottam a küldendő feljegyzést:

```
POST /cgi-bin/notes?cmd=add HTTP/1.0
User-Agent: Java1.0.2
Referer: http://orasun.hu.oracle.com:1000/cgi-
bin/notes?cmd=add
Accept: text/html, image/gif, image/jpeg, *; q=.2,
*/*; q=.2
Content-type: application/x-www-form-urlencoded
Content-length: 72
Fontos feljegyzés
Találka X-szel a Moszkva téren.
10 szál rózsát venni
HTTP/1.0 200 OK
Date: Thu, 22 May 1997 20:41:45 GMT
Allow: GET, POST
Server: Oracle_Web_listener2.1/1.20in2
Content-type: text/plain
Content-Length: 4
[4 bájt hosszú tartalom]
```

Bár a *NoteClient* osztályt arra szántam, hogy majd egy programka használja, jó programozási gyakorlat, hogy az egyes osztályainknak tesztelésére írjunk „főprogramot". Egy Jáva programban akárhány *main* lehet — osztályonként egy —, az elindításnál úgyis meg kell mondanunk, hogy melyiket hajtsa végre a virtuális gép. A program használatát három lehetséges parancsával mutatom be:

```
java NoteClient subjects java NoteClient fetch 4
java NoteClient add „Új fejléc" „Ez itt az új
```

```
megjegyzés"
```

Remélem, a program magától értetődő:

```
public static void main (String[] args)
{ NoteClient nc = new NoteClient
(„http://orasun.hu.oracle.com:2000/cgi-
bin/notes");
if (args[0].equals(„subjects")) {
Vector v = nc.getSubjects();
if (v != null) { Enumeration subj = v.elements();
while (subj.hasMoreElements()) {
System.out.println((String)subj.nextElement()); }
} else System.out.println(nc.getLastException());
} else if (args[0].equals(„fetch")) {
String note =
nc.getNote(Integer.parseInt(args[1]));
if (note != null) System.out.println(note);
else System.out.println(nc.getLastException());
} else if (args[0].equals(„add")) {
if (nc.addNote(args[1], args[2]))
System.out.println(„Note added.");
else System.out.println(nc.getLastException());
} else System.out.println(„Wrong command!"); }}
```

Mi történik a kiszolgálón?

A Web-kiszolgáló arra képes, hogy ha bizonyos, konfigurálható — esetünkben a /cgi-bin — könyvtárra hivatkozunk a kérésben, akkor az ott található futtatható programot elindítja. Legalábbis a Unix rendszereknél a program elindításakor a környezeti változókban megkapja a hívás fontosabb paramétereit, például:

```
HTTP_ACCEPT=image/gif, image/jpeg, image/pjpeg, */*
HTTP_ACCEPT_CHARSET=iso-8859-1, *, utf-8
HTTP_ACCEPT_LANGUAGE=en
HTTP_AUTHORIZATION=
HTTP_CONNECTION=Keep-Alive
HTTP_HOST=orasun.hu.oracle.com:1000
HTTP_REFERER=
HTTP_USER_AGENT=Mozilla/4.0b4 [en] (Win95; I)
PATH=/opt/java/bin:/oracle/app/oracle/product/7.3.2
/bin:/usr/bin:/
PATH_INFO=
PATH_TRANSLATED=
QUERY_STRING=cmd=fetch&ix=3
REMOTE_ADDR=193.9.149.249
REMOTE_HOST=
REMOTE_IDENT=
REMOTE_USER=
REQUEST_METHOD=GET
SCRIPT_NAME=/cgi-bin/notes
SERVER_NAME=orasun.hu.oracle.com
SERVER_PORT=1000
SERVER_PROTOCOL=HTTP/1.0
SERVER_SOFTWARE=Oracle_Web_listener2.1/1.20in2
```

Programunk számára a legfontosabb ezek közül a *QUERY_STRING,* amely az URL végéhez csapott paramétereket tartalmazza. Amennyiben a kliens POST módszert használt, a program a szabvány inputon keresztül olvashatja a kérés törzsét.

A CGI program a szabvány outputra küldheti válaszát, ezt a Web-kiszolgáló visszajuttatja a klienshez. A szerver azt a szívességet is megteszi nekünk, hogy összeállít egy szabályos válaszfejlécet, amelynek egyes mezőit azért felülbírálhatjuk. Ha a program elején olyan szöveget írunk, amely megfelel egy HTTP válaszfejlécnek, ezt használja, ha nem, akkor az általa megfelelőnek tartottat.

A CGI program

Programunk egyetlen *notes.txt* állományt használ a jegyzetek tárolására. Itt az egyes fejléceket tartalmazó sorokat a "——" karaktersorozat jelöli, a feljegyzés törzsét a következő sorok foglalják magukban.

A szerveroldalon futó programot tetszőleges nyelven megírhatjuk. Manapság a legdivatosabb talán a *Perl* — egy nagyon hatékony, interpretált szövegfeldolgozó nyelv —, de természetesen Jávában is dolgozhatunk. Én most az egyszerűség kedvéért a Unix alapparancsnyelvét, az *sh*-t, ún. *Bourne shell*jét használom. Aki esetleg nem értené, ne bánkódjon, elég annyit észrevenni, hogy milyen rövidke a program. Persze az *sh* mellett bevetettem még néhány Unix segédprogramot, mint az *awk* vagy a *sed.*

```
#!/bin/sh
```

Elágazás a QUERY_STRING környezeti változó tartalma alapján:

```
case $QUERY_STRING in cmd=subjects)
```

A fejlécekre van szükség. A válaszban visszaküldött információ típusa „szimpla szöveg" *(text/plain),* ezt külön ki kell írnunk, ellenkező esetben *text/html* lenne. Persze ezúttal ez sem okozna bajt, a kliensoldali Jáva program ugyanis csak az átküldött bájtokkal törődik. Szerencsére az üzenet hosszát itt sem kell számolgatnunk, ezt megteszi a szerver helyettünk:

```
echo „Content-type: text/plain\n"
```

Az *awk* program a *notes.txt* állományból csak azokat a sorokat, és azoknak is csupán a végét (a második szótól kezdődően) írja ki, amelyek kötőjelekkel kezdődnek:

```
awk ' /-----/ { for (i = 2; i <= NF; i++)
printf(„%s „, $i); print „" } ' notes.txt ;;
cmd=add)
```

Új feljegyzés érkezett. Az első sort beolvassuk, elé rakjuk a kötőjeleket, és hozzáírjuk az állományhoz. Aztán ciklusban addig olvasunk és írunk egy-egy sort, amíg tart a bemenet:

```
read line
echo „----- $line" >> notes.txt
while [ x = x ] do
read line if [ $? -ne 0 ] ; then break fi
echo „$line" >> notes.txt done
```

Végezetül udvariasan visszaküldünk egy „OK"-t, nem mintha a kliens törődne vele, de szükséges, hogy a szerver helyesen lezárja a kapcsolatot:

```
echo „Content-type: text/plain\n"
echo „OK!" ;; cmd=fetch*)
```

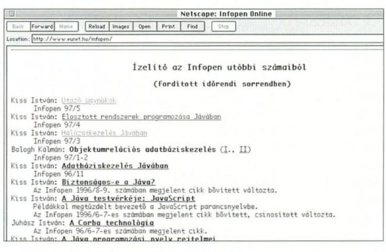
Egy adott sorszámú feljegyzést kérnek. Az *awk* parancs az *i* változóban számolja azokat a sorokat, amelyek kötőjelekkel kezdődnek. Akkor ír majd ki, ha a számláló az adott értékű:

```
echo „Content-type: text/plain\n"
awk ' $1 ~ /-----/ { i = i + 1; next }
i == ix ' ix=`echo $QUERY_STRING | sed
s/cmd=fetch&ix=//` notes.txt ;; *)
echo „Content-type: text/plain\n\n"
echo „Wrong parameters!" ;;esac
```

Ezúttal ennyire futotta. Arról pedig, hogy hogyan kell ugyanezt Jávában megírni, majd legközelebb.

KISS ISTVÁN

Helytakarékossági okokból a programszöveg írásmódján helyenként módosítottunk, a szerző által készített változat a http://www.eunet.hu/infopen *címen található.*



AXIS: „Három évre előre tervezünk"

Sybase-stratégia 2000-ig



Három évre szóló disztribútori szerződést kötött a Sybase céggel az AXIS Kft., amely így 2000-ig képviseli a Sybase programtermékeket Magyarországon. Az AXIS az utóbbi két-három évben fokozatosan áthelyezte tevékenységének súlypontját a Sybase termékekre, és tavaly ősszel a hivatalos Sybase Oktató Partner címet is megszerezte. Újabban a cég kelet-európai működését a prágai leányvállalat irányítja. Ily módon a korábbinál több támogatást kap az AXIS, amelynek ügyvezető igazgatója, *Dénes László* foglalta össze a közeljövő cégstratégiáját, a Sybase termékek értékesítésében várható pozitív változásokat.

Prioritások

A magyarországi piacra vonatkozó prioritások Dénes László szerint a következők:

- adatáruházak/adatraktárak, a Sybase újszerű megközelítésében;
- az SQL AnyWhere adatbázis-kezelőt tartalmazó, nagyon versenyképes áron kínált Workplace Database bevezetése;
- PowerBuilder fejlesztőeszközök további elterjesztése;
- Internet/intranet megoldások népszerűsítése a Sybase Adaptive Component Architecture elemgyűjteményével, amelyből bárki kiválaszthatja a testre szabott megoldást, akármilyen hardver- és/vagy szoftverkörnyezet esetén.

Négyen a középpontban

Jelenleg négy fő területre koncentrálnak. Az egyik a klasszikus termékdisztribúciót, a másik a Sybase termékek műszaki kiszolgálását, a konzultációt, felhasználói támogatást és verziókövetést öleli fel. A harmadik, legalább ilyen fontos terület a tanfolyamok szervezése. A kft. budapesti Oktató Centrumában az idén indult be a rendszeres — negyedéves tervek alapján megvalósuló — tanfolyami képzés, amely már eddig is sok sikert könyvelhetett el. Negyedikként az alkalmazásfejlesztést kell megemlíteni, aminek lényege, hogy az AXIS a Sybase termékekre épülő rendszerfejlesztésekhez a felhasználók igénye szerint biztosít szakembereket (rendszerszervezőket, adatbázis-tervezőket stb.). Az sem gond, ha nem száz százalékig homogén Sybase termékekre épülő fejlesztésekről van szó, ugyanis az összes Sybase eszköz nyitott alapfilozófiája lehetővé teszi a más eredetű programtermékek támogatását. Példa erre a PowerDesigner CASE eszköz, amely többféle fejlesztőeszköz számára is képes a megfelelő sémákat generálni, vagy a PowerBuilder alkalmazásfejlesztő, amelyik a piacon elterjedt valamennyi jelentős adatbázis-kezelőt támogatja.

Az indirekt csatorna erősítése

Egyre több felhasználó közvetlenül a disztribútortól szeretné megkapni a termékeket, szolgáltatásokat és a legújabb Sybase technológiákat. Az AXIS ez év május közepétől kétféle partneri szerződést ajánl: ún. kereskedelmi alkalmazásit fejlesztőcégek, és rendszerintegrátori konstrukciót rendszerintegrátorok számára.

A következő három évben a hangsúlynak át kell tevődnie az indirekt értékesítésre, 1999 végére az AXIS-nak Sybase leányvállalatként kell majd működnie. A partnerek számára is hasznos lesz, ha a disztribútor egyre inkább az általános piaci marketingre és az ő támogatásukra

koncentrál. Már jelenleg is 8-10 olyan hazai cég van, amelyek a Sybase termékek körében tanácsokat tudnak adni a felhasználóknak. Eddig mintegy 10-15 céggel alakított ki hosszabb távú kapcsolatot az AXIS Sybase ügyben. Dénes László úgy gondolja, hogy 1997 végére 8-10 szerződött partnerük lesz a fejlesztőcégek és rendszerintegrátorok között. Rá egy évre ez a szám 15-20, 1999 decemberére pedig 25 körül várható. A fejlesztőcégek közül már ma is többen Sybase eszközöket használnak a kínált alkalmazásaikba beépítve.

Felfutó oktatás

A Sybase stratégiája, hogy országonként csak egyetlen cég képviselheti disztribútorként termékeit; az oktató partnerek számára nem létezik ilyen megkötés. Az AXIS szívesen venné, ha több cég is teljesítené azokat a minőségi követelményeket (szakemberek tudása, oktatóhely berendezése), amelyek alapján megszerezhetik és használhatják a Sybase Oktató Partnere címet.

„ImpactNow!"

1996 közepe óta új menedzsmenttel működik a Sybase, amelynél a korábbi szétaprózottságot és a termékek fokozat nélküliségét felváltotta a mindössze 6-7 kulcsterületre, kulcstermékre való koncentrálás. Az eredmények pozitívak: 1997 első negyedévében a cég nyereséget produkált, és egy sor új termék jelent meg ebben a félévben. Ennek a változásnak az üzenetét vitte szét a világba a Sybase a májusban lebonyolított „ImpactNow!" nevezetű rendezvényeivel. Amerikában minden nagyvárosban, Európában pedig huszonkét helyen tartottak nemzetközi szakmai napot.

A Sybase magyarországi elszántságát jelezte, hogy Prágán kívül (ahol kelet-európai leányvállalata működik) csak Budapesten szervezett ilyen rendezvényt a kelet-európai régióban. Az amerikai vállalat a technológiai irányok mellett három területre kihegyezve mutatta be eredményeit, megoldásait, ezek: a most megjelent Adaptive Component Architecture, az adatáruházak és az Internet.

Közös akciók

A jövőben az AXIS erősíteni szeretné a Magyarországon is meghatározó nagy hardverforgalmazókkal kialakított kapcsolatait. Az elmúlt néhány évben ötven olyan nagyobb Sybase installáció történt, amely az AXIS nevéhez fűződik. Ezek révén kapcsolatba kerültek többek között a Digitallal, a HP-vel, az IBM-mel és a Sunnal. Most azt remélik, hogy az alkalomszerű közös akciók szorosabbra fonják majd ezeket a kötelékeket. A május 26-i ImpactNow! rendezvényt például világméretű megállapodás alapján a Sun támogatta (előadással is részt vett a budapesti szakmai napon).

Műhely és klub

Dénes László szerint az AXIS a jövőben időről időre különféle rendezvényeket fog tartani: tavasszal és ősszel egy-egy több száz fős szakmai napot, amely tökéletesen beleillik a Sybase világméretű marketingjébe. Idehaza szeptember-október táján két újdonságot is bevezetnek: egyrészt a fejlesztőcégeket szeretnék mind alaposabban megismertetni a Sybase technológiákkal az egy-egy termékhez kapcsolódó műhelyfoglalkozások segítségével; másrészt havi, kéthavi rendszerességgel működő Sybase–Powersoft felhasználói klub indítását tervezik.

Kovács Attila



Stabil pilléreken a RISC jövője

Áprilisban a Kanári-szigeteken, Tenerifén rendezték meg az évente szokásos IBM Power Academyt, amely az RS/6000 számítógépcsaládról, illetve ezen belül az IBM PowerPC-s stratégiájáról adott számot. A több száz résztvevő között az IBM üzleti partnerei is jelen voltak. *Szabó Balázs*, az IBM Magyarországi Kft. termékigazgatója, valamint két Tenerifén járt szakember, *Beer György* RS/6000 termékmenedzser és *Csurgai Gábor*, az IBM-partner OpSys Kft. ügyvezetője nyilatkozott lapunknak. Olyan kérdésekről beszélgettünk, hogy milyen pilléreken nyugszik a PowerPC technológia jövője, mire alapozza az IBM szoftvertechnológiáját, milyen előnyöket jelenthetnek a versenytársakkal szemben minden téren megnyilvánuló komoly, saját fejlesztések, miben rejlik az IBM technológiamarketing lényege.

A Power Academy fényében hogyan látják a PowerPC platform jövőjét?

Sz. B.: A RISC technológia stabil fennmaradásáról tanúskodik, hogy nemcsak számítógépeinkben, hanem úgynevezett beágyazott rendszerekben, vezérlőkben is egyre szélesebb körben alkalmaznak ilyen processzorokat a világon. Gondolok itt például a digitális kamerákra, televíziós settop boxokra vagy az otthoni számítógépek (pl. Nintendo) többségében előforduló RISC processzoros megoldásokra. Nagyon széles és egyre bővülő az egyszerűsített RISC processzorok alkalmazási köre, amelyek közül az IBM Network Stationben is ilyeneket találunk. A nagy megbízhatóságú, sokfelhasználós szerverekben ez a technológia tovább él, a beágyazott rendszeres alkalmazások pedig biztosítják a gazdaságos gyártáshoz szükséges mennyiséget.

B. Gy.: Tenerifén is meggyőződtünk arról, hogy a PowerPC technológia jövője sokkal világosabb, mint bármely más vezető processzortechnológiáé. E téren *Mark Bergmanra*, az RS divízió igazgatójára vagy *John Dodsonra*, a fejlesztések egyik meghatározó szakemberére is hivatkozhatom. Mindketten Magyarországon is tartottak már előadásokat az IBM-partnereknek. A nagyon stabil egyenszilárdságú rendszerek sikeres jövőjét nemcsak az állandó processzorsebesség-növelés, hanem a vele párhuzamosan zajló egyre gyorsabb memória-alrendszerek kifejlesztése is jelenti.

A fejlődés következő lépcsőfoka a 64 bites technológia lesz. Ennek lényege az RS/6000-esekkel kapcsolatban, hogy egyetlen fázisban fog megvalósulni a hardver, szoftver, adatbázis-kezelés stb. vonatkozásában. Most is elhangzott: mind a PowerPC, mind a Power2 technológia szuperskalár RISC technológia, ami biztosítja az egy órajel alatt több utasítást végrehajtani képes processzorműködést.

Cs. G.: A Power Academy rendezvény is bizonyította, az IBM-nél egyre növekszik a partnerek jelentősége, egyre erősödik a technológiamarketing szerepe. Ennek is betudható, hogy a Kék Óriás kezdi leépíteni saját eladási csatornáit, arra koncentrálva, hogy minden téren a műszaki támogatást és a technológiai fejlesztést helyezze előnybe. Az IBM azért tud továbbra is vezető maradni, mert az összes élen járó technológiát egyszerre, a lehető legmagasabb színvonalon ötvözve, integrálva képviseli termékeiben, megoldásaiban.

Merre haladnak az RS/6000-esek, mik a leglényegesebb újdonságok?

B. Gy.: Az RS/6000, illetve a vele szállított Internet szoftverek lehetővé teszik, hogy olyan alap-Internet alkalmazás jöjjön létre, amelyre támaszkodva bármilyen Internet technológiát meg lehet valósítani. A kommunikációs technológiában az ISDN, a 100 Mbites Ethernet és az ATM technológia már a PCI buszos gépekben is megjelent. Újdonság az Ultra SCSI buszadapterek elérhetősége. Az AIX új verziója az összes RS gépet támogatja, tartalmazza a Lotus Domino egyprocesszoros változatát, Internet szerverprogramokat, s a Network Station szoftveres támogatása az operációs rendszer részévé vált.

Cs. G.: A hardver áráért a felhasználó megkapja az operációs rendszert, továbbá olyan kiegészítő programokat, amelyekkel játszva egy működő Web-szervert lehet létrehozni. Ezt ma még eléggé kevesen tudják a piacon nyújtani. Egységes csomagként jelenik meg minden olyan eszköz, amelyből akár egy intranetes technológiára épülő belső hálózati rendszer is kialakítható. Az IBM komoly erőforrásokat állít csatasorba azért, hogy saját maga „hangolja át" megoldásait, szoftverkészletét az intranetes technológiákra. Az is megkülönbözteti versenytársaitól, hogy az alap- és a kiegészítő eszközöket egységesen maga fejleszti tovább.

Nagyon fontos az IBM nevéhez fűződő Secure Electronic Transaction protokoll, amelynek segítségével biztonságosan lehet kereskedni az Interneten keresztül.

B. Gy.: Az IBM egyszerű Internet technológiája a Net.Data, amellyel nagy adatbázisok vihetők az Internetre. Nagyon jó példa arra, hol tart e téren a cég, hogy az amerikai kormányzat megbízásából az összes amerikai szabványt ezzel a technológiával helyezte ki a Webre.

Mit hoz a jövő, különös tekintettel az Intel-alapú rendszerek megújuló kínálatára?

Sz. B.: Látható, hogy az új technológiák egy jó ideig feltétlenül együtt fognak élni a meglévőkkel — mindez az installált bázis nagysága miatt van így. Az Intel és a RISC közötti különbség vonatkozásában elmondható, hogy az Intel technológiával kialakított szerverek még sokáig nem abban a kategóriában kerülnek felhasználásra, mint ahol a RISC technológiával készültek. A PowerPC-vel épült AS/400 és RS/6000 jóval nagyobb felhasználószámot, magasabb szintű megbízhatóságot eredményez, mint amit az Intel technológia ma kínál.

Fontos hangsúlyozni, hogy szoftvertechnológiánkat egyre inkább a mindent átható szabványosság jellemzi. A más platformokon való működés lehetősége egyre nagyobb mértékben fennáll, nem kell tehát feltétlenül IBM hardvert telepíteni meglévő szoftverek alá. Megbizonyosodhattunk arról, hogy a skálázhatóság, az egyenszilárdságú rendszerek és komponensek nem üres szlogenek az IBM-nél, mint ahogy a „Let's net together!" jelmondatát is komolyan gondoljuk, hiszen minden megoldásunkba csakugyan beleértendő, hogy azok hálózatban együtt tudjanak működni.

Cs. G.: Tenerifei tapasztalatom, hogy a klasztertechnológia valóban megbízhatóan működik az IBM-nél, az operációs rendszerek terén a termékeik nagy skálázhatósági tartalékokkal rendelkeznek. Másokhoz viszonyítva a megbízhatósági szempontokat az alapszoftvereknél sokkal előbbre helyezik, egy bevezetési periódus alatt szinte mindig élesben lehet működtetni a rendszereket.

Tenerifén engem leginkább az NC-technológia ragadott meg, amit az IBM a zászlajára tűzött; meggyőzőnek találtam a Network Stationnel kapcsolatos érveiket is. Náluk mindaz rendelkezésre áll, ami a szerver oldalon ehhez szükséges, s úgy gondolom, óriási erőforrásokat fordítanak erre a technológiára.

Hiszek az NC-technológia hazai elterjedésében is, mert csak így látom biztosítva a műszaki támogatás, a megbízhatósági szempontok, valamint a biztonságos rendszerek kritériumainak érvényesülését Magyarországon.

KOVÁCS ATTILA

Kötetnyi információ alig száz oldalon

A megszokottól kissé eltérően rögtön az összegzéssel kezdem: az a viszonylag szűkebb réteg, amelyiknek ajánlóm szól, biztosan nem pazarolja értékes idejét, ha figyelmesen végigböngészi a könyvet. Pazarlásról már csak azért sem beszélhetek, mert a szerző igen tömören (alig 100 oldalon!) tárgyal egy olyan témakört, amelyről vaskos köteteket is gond nélkül lehetne írni.

Jutasi István:
Az Internet felépítése és működése
Műszaki Könyvkiadó, 1997
ISBN 963 16 1253 8

S most vissza az elejére: kinek is szólhat ez a könyv? A profi Internet guruknak semmiképp sem, hiszen ők valószínűleg álmukból felverve is sokkal többet tudnak a különböző internetes technikai-szervezeti megvalósításokról, mint amennyi ebben a kötetben olvasható. Az átlagos felhasználó is nyugodtan visszateheti a polcra — ő meg aligha tudná hasznosítani az itt talált műszaki információkat. Ám ne feledkezzünk meg azokról az informatikai és számítástechnikai szakemberekről, akik még sohasem kerültek komolyabb kapcsolatba az Internettel, de a változó követelményekhez igazodva ezt már nem halogathatják. Sok esetben ugyanis nekik kell dönteniük, javaslatot tenniük cégük számára arról, hogy miként érdemes kapcsolódni a világhálóhoz. Nekik jelenthet nagy segítséget ez a kiadvány.

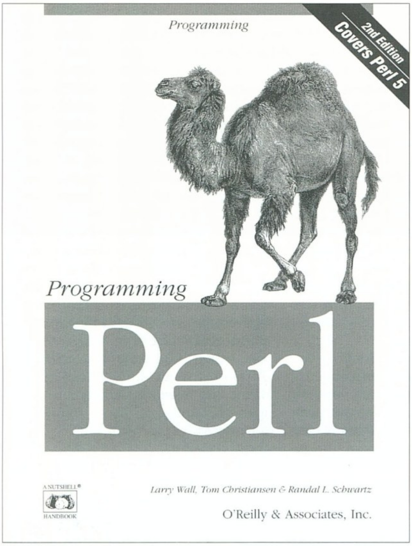
Jutasi István — a könyv méreteihez képest — meglepően részletesen tájékoztat minket az Internet működésével kapcsolatos műszaki alapokról. A távbeszélő-világhálózatok elvi vázlatából kiindulva, a csomagkapcsolt adathálózatokon keresztül eljut egészen az Internet technológiáig. Ezt igyekszik lényegretörően tenni, olyannyira, hogy a hálózati témákban járatlanabb olvasónak bizony néha erősen koncentrálnia kell ahhoz, hogy el ne veszítse a fonalat. Persze ezt a fejezetet akár át is lapozhatják azok, akik nem szeretnék részletesebben megismerni az Internet fizikai valóját, ám véleményem szerint megéri rááldozni egy kevéske időt az itt találhatók megértésére.

A könyv a technológiai ismertetésen kívül bemutatja, miként kapcsolódik kicsiny hazánk a hálózathoz, és összegyűjti a magyarországi Internet-szolgáltatók neveit. A szerző — meglehetős tömörséggel — összefoglalja a különböző Internet-szolgáltatásokat is, átfogó képet adva lehetőségeinkről.

A világhálózathoz való csatlakozás mindenkitől megköveteli, hogy tisztában legyen az alapvető törvényi és szokásjoggal. A mű természetesen ezekre a témakörökre is kitér, ismertetve az Internethez kapcsolódó szerzői jogokat, valamint a hálózat etikai kódexét, a Netikettet. Jutasi a biztonsági kérdéseknek is szánt egy fejezetet, amelyben — ha igen vázlatosan is, de — foglalkozik a legnagyobb problémákkal.

Csapó Attila

Tevék, lámák és egyéb állatfajták



L. Wall, T. Christiansen, R. L. Schwartz: Programming Perl O'Reilly & Associates, 1996 (Second Edition) ISBN 1-56592-149-6

E havi recenziónkat (melyhez az alapanyagot megint csak a *Software Station* könyvespolcáról emeltük le) ezúttal a klasszikusok tanulmányozásának szenteljük. (Nem, ne arra a bizonyos vörös-arany kötéses sokkötetes ideológiai alapvetésre gondoljon az olvasó — amiről szó lesz, az inkább a Brehmre üt, legalábbis kívülről, ugyanis a szakmában legendás O'Reilly kiadó egyik kötetére csaptunk le mohó kezekkel; e könyvek két fő jellegzetessége pedig egyrészt a kiváló belbecs, másrészt a borítót díszítő, szintén igényes állatgrafika.) Fent nevezett klasszikus címlapján egy rokonszenves, púpját (pardon, másságát) büszkén vállaló teve feszít, de mint majd kiderül, nem csak ezért szeretjük... A híresneves „Camel Book", avagy ahogy a kevésbé tájékozott néprétegek mindközönségesen emlegetik, a „Programming Perl" (annak is a második kiadása) került ugyanis kantárvégre. Jelen sorok írójának ez azért különösen becses zsákmány, mert volt szerencséje részt venni a könyv második kiadásának ünnepélyes bemutatásán Chicagóban, ahol volt mindenféle dínomdánom, jégből faragott életnagyságú teve az ínycsiklandozó finomságokkal megrakott svédasztalon, Valódi Életnagyságú Dedikáló Szerzők, több száz résztvevős teveverseny (a tevék a nálunk is ismert, kivált szilveszterkor népszerű, lejtős felületen lekocogó szerencsemalacok púpos rokonai voltak) — szóval, volt ott minden, csak épp az ünnepelt könyv nem, mert az bizony az első percekben elfogyott, így a dedikációk is csak a (természetesen szintén tevés) pólókra jutottak... Ekként belátható, hogy a könyv megszerzése a szegény recenzor (brrr...) számára már-már mániás kényszerré vált, de mint a továbbiakban bízvást kiderül, nem ok nélkül.

Futólag már volt szó e hasábokon a Perlről, mely üstökösszerű karriert futott be a parancsértelmező jellegű nyelvek között, s e karriernek még közel se látszik a vége. Mi is hát ez a nyelv? Már maga a név is többrétegű, a hivatalos (értsd: nyakkendősöknek szóló) feloldás szerint a Practical Extraction and Report Language rövidítése, amiben van is némi igazság, a Perlt ugyanis alapvetően a sed és az awk mintájára, szöveges információk (tovább)feldolgozására találták ki. Az Igazi Programozók természetesen a nyelv megalkotója *(Larry Wall)* által sugallt másik feloldásra esküsznek, nevezetesen: a Perl nem más, mint egy Pathologically Eclectic Rubbish Lister. Hááát, hogy is mondjam? Van benne igazság... A Perl bevallottan a különböző shellek, a sed, az awk és a C leghasznosabb funkcióinak összegyúrásából született, s noha az eredmény kétségkívül igazolta a szerzőket, az biztos, hogy nyelvszépségversenyen nem túl nagy eséllyel indulhatna. Jobb érzésű és akadémikus beállítottságú szakemberek, akik egy változót csak a típusdefiniálás és inicializálás hagyományos rituáléja után szólítanak elő a nemlét mezejéről, már a hagyományos Unix shellek láttán is megszaggatták ruhájukat és hamut szórtak a fejükre; nos, a Perl számukra a szélütés és az epileptikus rohamok veszélye miatt kifejezetten ellenjavallt. A Perlben való elmélyülés során az embernek óhatatlanul a klasszikus vicc jut eszébe, miszerint a teve az a ló, amelyet egy bizottság tervezett, s ha az is eszébe jut, a teve se egy kimondottan szexepiles állat, óhatatlanul az a rossz érzése támad, nem ok nélkül került pont ennek az O'Reilly-kötetnek a címlapjára.

De ne legyünk igazságtalanok, a hasonlat sántít, belül igencsak öszszefogott és átgondolt nyelvről van szó, mely megjelenése óta újabb és újabb funkciókkal egészült ki, s jelenlegi ötös verziója már a legtöbb általános célú programnyelvvel felveszi a versenyt. A hagyományos szövegfeldolgozási funkciók kiegészültek debuggerekkel, optimalizálókkal, fordítóprogramokkal, objektumorientált könyvtárakkal és szintaxisvezérelt szerkesztőkkel egyaránt, s számos olyan területen alkalmazzák már, ahol ez korábban elképzelhetetlen lett volna (ez persze a hamuszóró és ruhaszaggató szakemberek számának elkerülhetetlen csökkenésével jár, de hát a fejlődésnek óhatatlanul vannak mellékhatásai). Perl scriptekkel találkozhatunk már CAD/CAM alkalmazásokban éppúgy, mint DNS szekvenciavizsgáló programokban, vagy Web szerverekben (a legrövidebb, alig egyoldalnyi kódot tartalmazó Web szerver szoftvert is Perlben követték el). A legnagyobb lökést azonban kétségkívül éppen ez, a Web megjelenése adta. Ma a legtöbb Web szerveren Perl scriptek végzik el a háttérben a CGI scriptek által megszabott „piszkos munkát". Jó okkal: a Perl gyors, flexibilis, robusztus és hordozható, kimondottan azokra a funkciókra tervezett, amikre e környezetekben használják.

Egy közkeletű félreértés eloszlatására: a Perl nem tisztán interpretív jellegű nyelv, a forrást először egy közbenső formára fordítja, s azt hajtja végre az értelmező; ez a hagyományos parancsértelmezőknél gyorsabb végrehajtást eredményez, ugyanakkor megmarad a gyors fejlesztés-futtatás-javítás életciklus, ami ideálissá teszi a rapid prototyping jellegű programozási feladatok ellátására is. A nyelv legújabb verziói „csont nélkül" bővíthetőek tetszőleges funkciójú új modulokkal, és zökkenőmentesen integrálhatóak Perl programokba más nyelvű modulok (s gyakran fordítva is).

A hordozhatóság sem puszta szlogen, a Perl (már persze a befogadó operációs nyelv nyújtotta kereteken belül) ugyanúgy elfut Unix, mint Windows, Macintosh, VMS vagy DOS alatt. S végül utoljára, de nem utolsósorban, a legjobb unixos hagyományokhoz hűen a Perl is ingyenesen hozzáférhető, számtalan kiegészítő programmal együtt. Noha a legtöbb Unix verzióban immár hivatalból megtalálható a Perl, s a Webbel kapcsolatos CD-s programgyűjtemények se teljesek nélküle, a legfontosabb tájékozódási pontokat mindenképpen érdemes megemlíteni. A Perl-hívők egyre gyarapodó táborának Kába-köve nem meglepő módon a *http://www.perl.com/perl* cím, ahol naprakész információk mellett a teljes forráskód és az öszszes említésre méltó perles alkalmazás fellelhető. A nemrég alakult, nem profitcélú Perl Institute anyagai a *http://www.perl.org* címen találha-

tóak, az elektronikus újságcsoportok közül pedig minden valamirevaló site tükrözi a *comp.lang.perl.** rovatokat. Magát a kódot és az egyéb Perl anyagokat a *CPAN (Comprehensive Perl Archive Network)* archívum tartalmazza, ezt számos helyen tükrözik a világban (többek közt az *ftp.kfki.hu* gépen is), a *http://www.perl.com/CPAN* címre lépve kiválaszthatjuk a legközelebbi tükröt (a cím végére még egy / jelet bigygyesztve a szerver automatikusan választ egyet, nem feltétlenül helyesen).

A Perl után nézzük magát a könyvet! Aki ismeri az O'Reilly sorozatot, annak elég, ha annyit mondok, olyan, mint a többi. Azaz nagyon jó. (Épeszű kritikus ugyan tartózkodik a túlfűtött érzelemnyilvánításoktól, de kivételek mindig vannak, s az én számomra mindig is az O'Reilly könyvek jelentették az etalont, mind témaválasztásban, mind tipográfiában vagy terjedelemben, hogy magáról az érdemi információról ne is beszéljünk. Lakberendezési álmaimban mindig ott szerepel egy legalább két méter hosszú, a teljes O'Reilly sorozatot tartalmazó könyvespolc, de rendszerint ilyenkor szoktam felébredni...) A „Programming Perl" bevallottan referencia-kézikönyvnek íródott, de a Perl valamelyik ősének ismeretében (sh, csh, sed, awk, C) már önállóan is emészthető olvasmány.

Az első fejezet viszonylag részletes kóstolót nyújt a nyelvből, s utána már a mélyvíz következik; aki ennél kíméletesebb szoktatásra vágyik, annak bízvást ajánlhatjuk a *„Llama Book"*-ot, azaz az ugyancsak O'Reilly-kiadású *„Learning Perl"* című könyvet.

A második fejezet a nyelv belső alapfogalmait tekinti át, a különféle adattípusokat, változókat, függvényeket és szubrutinokat és modulokat. A harmadik fejezet a beépített függvények részletes ismertetését tartalmazza, a negyedik a változóhivatkozásokat és beágyazott adatstruktúrákat mutatja be, az ötödik pedig a Perl objektumorientált jellemzőit, a package-ek és modulok használatát. (Csak emlékeztetőképpen, az objektumorientáltság az a bűvszó, aminek hatására a jóérzésű stb. szakemberek szélütéses tünetei némileg enyhíthetőek.)

A hatodik fejezet a „másokhoz" illesztés lehetőségeit tekinti át, a másokba beleértve az adott gépen futó egyéb Unix programokat épp úgy, mint a hálózatos alkalmazásokat, illetve beágyazott más nyelvű (elsősorban C és C++) programmodulokat. A hetedik fejezet a standard Perl könyvtár funkcióival foglalkozik — használatát bátran ajánlhatjuk mindazoknak, akik nem éreznek leküzdhetetlen vágyat a kerék újrafelfedezésére.

A nyolcadik fejezet, afféle vegyes salátaként, a debugger ismertetése mellett számos jó tanácsot nyújt a kezdőknek és kevésbé kezdőknek egyaránt, mind a programok hangolását, mind a célszerű programozási stílust illetően.

Az utolsó fejezet a Perl diagnosztikai üzeneteit ismerteti, végül meglepően gazdag szószedet és a tárgymutató zárja a könyvet.

Mivel is fejezhetném be ezt a recenziót? Aki Perlt akar használni (és ilyenek, hál' istennek, egyre többen vannak, rendszergazdáktól a „neccmesterekig", oroszlánkörmüket próbálgató gimnazistáktól a szakma nagyjaiig), annak a „Tevekönyv" nem maradhat le a polcáról; aki pedig még nem ismeri a nyelvet, mindenképpen érdemes e hiányt pótolnia...

BARTÓK NAGY JÁNOS
JANOS@ILAB.SZTAKI.HU


NETWORX
TUDÁS
TAPASZTALAT
GYORSASÁG
MEGBÍZHATÓSÁG

TOPszáz topLÁZ

Bajnokok Próbája

Üdvözlet minden kedves TOPológusnak! A fenti cím alapvetően nem a hányatott sorsú szerepjáték-könyveknek eme minden eredetiséget és izgalmat nélkülöző kötetére utal, hanem arra a két dologra, amelyekről mostani jelentkezésemben írok. Az egyik természetesen a Top100 bajnokok megmérettetése, a másik az az embert próbáló feladat, hogy egy jelentősen ronda kinézet mellett még tudjunk gratulálni egy site tartalmához. Sajnos nálunk ez még szinte mindennapos, ezért a cikkem végén néhány rövid pontban leírom, mi szükséges ahhoz, hogy az értékelés „design" címszavánál 50%-nál jobbat mutasson a site.

Ígéret szép szó... Az RTV Újság pedig még mindig fent van a listán, tehát...

RTV Újság Online (7. helyen)
http://www.rtvujsag.hu/

APEH. Bah. Miniszterelnöki Hivatal. Bah. Elektronikus könyvtár. Bah. EZ az igazi közszolgálati website, hiszen mi fontosabb annál, mint hogy mi megy ma este a tévében?

Bár ez csak vicc volt, senki sem vitatja, milyen jelentős szerepet tölt be társadalmunkban a tévé. És mit ér magában a tévé, ha nem tudjuk, mikor mi megy?! Hát ezért van nekünk klasszikus RTV magazinunk (ja, abban még rádió is van...). Most már webesítve is. Szép. Mármint a koncepció és a tartalom. De a dizájn... Uhh. Tagadhatatlan, hogy jobb, mint az MTV-é, de igencsak hagy kívánnivalót maga után. Mivel azonban ez lassan már tömegjelenség (jó tartalom rossz külcsínben, sok, de nagyméretű és rossz minőségű grafika stb.), foglalkozzunk a tartalommal.

Aki ismeri az újságot, az tudhatja, hogy nagyjából mit találhat itt: rádió-, tévé- és műholdas műsorokat, vannak itt linkek az adókra (nem az APEH által definiáltakra... kuc-kuc), igaz, eléggé elbújtatva, aztán vannak linkek a már Neten is szóló rádióadókra, horoszkóp, Westel900-as Internet-iskola, MIDI-zenefájlok (kiváltani az újság zenei rovatát)... Szóval sok minden, az egész készítőjének, az InterNeon-nak az Infocenter nevű kiadványán (ami egy hangyányit azért jobban néz ki) még időjárás-jelentés is, de a legjobb mégis a letölthető egész heti műsorújság. Ha netán sokallnánk mind a telefonszámlát, mind az újságköltséget, hetente egyszer letöltjük az egész zip-archívumot, és egy hétre el vagyunk látva a „legszükségesebb" információval.

Legvégül a bajaim: 1. grafika — borzasztó; 2. interaktivitás: nulla (a műsorkeresésnél a legjobb az lehet, ha az ember bemegy az őt érdeklő csatorna lapjára, és a böngésző „find" gombját lenyomva... szörnyű, nem?). Mindent összevetve, a site használható, de nekem már amúgy is jár papírformában, és egy műsort itt megtalálni ugyanannyi, mint ott...

- Tartalom: 85%
- Koncepció: 50% (nem interaktív)
- Design: 40%
- Gyorsaság: közepes
- Összhatás: 60%
- Könyvjelzőt neki: nem rossz ötlet
- Browser: fejlettebb Navigator vagy Explorer

Mint azt múltkor megígértem, egy nem pont toplistás, de érdekes site is terítékre kerül:

Private Moon Productions
http://www.privatemoon.com/

Igényességért sajgó lelkemnek végre egy kis gyógyír! Aki nem tudná, hogy miről van szó (számítástechnikai világban nem is várom el), Pierrot, a hajdani fél-bohócarcú, egykor zeneszerző/előadó, ma producer/zeneszerző/lapkiadó-mindenes saját produkciós cége a Private Moon Productions. Aki ezt elolvasta, sejtheti, hogy a site legalábbis unalmas nem lesz — és bizony, érdekes és sokszínű site-tal állunk szemben. Van itt Pierrot Fan Club (nos, ez a legelvetemültebbek számára lehet csak izgalmas), a PMP által képviselt együttesek honlapja (Sipos F., Kartell stb.), a Display (expapír, most under construction netes magazin, Roland-hangszerekbe pistult zenészeknek), az Inventory és Trollbarlang fantasy/szerepjáték magazinok és erőforrások. No és persze Pierrot minden mennyiségben.

Ami tetszik: gyorsan töltődő, igényes képek; jól kigondolt, átlátható website; nagy információtartalom. Ami nem tetszik: lassan töltődő, felesleges képek; zavaró, az olvasást néha lehetetlenné tevő hátterek; a nagy under construction-hangulat.

Igényesség szempontjából ez a site kb. Makó–Jeruzsálem szinten áll a (lassan már etalonszámba menő) felkavaró MTV-site-tól. De! Manapság már nem Jeruzsálem a mérvadó. Ma kb. New York magasságában van a léc.

- Tartalom: 90%
- Koncepció: 80%
- Design: 80%
- Gyorsaság: közepes
- Összhatás: 80%
- Könyvjelzőt neki: akit érdekelnek a site témái
- Browser: fejlettebb Navigator vagy Explorer

Zárás előtt akkor jöjjön egyrészt a szokásos gyorshalmaz, másrészt az elején beígért tipphalmaz:

HLE — Hungarian Link Exchange (*http://hle.isys.hu/*): a Top100 szellemében fogant, de egészen mást megvalósító szolgáltatás. Újfent iSYS Hungary (most már P-K4) produkció. És úgy tűnik, lassan olyan sikeres is. Mi is ez? Alapvetően ingyenes site-közti reklám. Aki konkrét példára kíváncsi, keresse a külföldi lapokon a Link Exchange logókat. Ez jobbára ugyanaz, magyarra portolva.

www.hullamfront.com (*http://www.hullamfront.com/* vagy *http://www.hullamfront.hu/*): becses kis személyes honsite-om, június elsejétől minden kedves érdeklődőt satöbbi.

És a stílustanácsok:

- a fehér, egyszerű háttér *szép*
  ... de ha már mindenképpen kell valami háttérgrafika, az oldalba legalább ne zavarjon bele (fekete alapon fekete szöveg)
- kevés grafika, ügyesen elhelyezve, kisméretű fájlokban
- interaktivitás
- kerüljük a felesleges Java/Javascript cuccokat
  ... de ha már mindenképpen kell, legyen inkább JavaScript
- az animált GIF-ek lehetnek figyelemfelkeltőek, szépek, érdekesek és ocsmányak. A legutolsó kategóriát próbáljuk meg elkerülni... FIGYELEM! az animált GIF sajnos hosszabbá teszi a fájlt...
- kerüljük az előre gyártott grafikák alkalmazását, próbáljunk meg sajátot alkotni
- legyen egységes kinézete a sitenak
- használjunk színharmóniát is az egységesség megteremtéséhez
- könnyen lehessen navigálni
  ... és ha már mindenképpen FrontPage-dzsel kell dolgozni, legalább azokat az idegesítő kódlap-azonosítókat szedjük ki a végén, mert a site gyakran olvashatatlan lesz más browsereken...

Ennyi fért ebbe a hónapba. Jövő hónapban ugyanitt, jobbára ugyanígy, plusz egy interjú a Top100 és a HLE alkotójával, ha összejön.

Boldog és felelőtlen szörfözést kívánok!

ROVOTT VEZETŐ: KE.AN
(KEAN@HULLAMFRONT.COM)

Hálózatokról az Ifabo ürügyén

„Let's Net Together"

tv3

Megszokhattuk, hogy manapság a nagy számítástechnikai seregszemlék, ha úgy tetszik, legnagyobb szenzációját a hálózati számítástechnika, a különböző inter- és intranetes megoldások, s az ezek nyomán vagy éppen erre a kihívásra válaszként piacra kerülő, egyre nagyobb háttértárolókkal felszerelt, mind gyorsabb, sokkal inkább felhasználóbarát hardvereszközök jelentik.



Közvetlen kapcsolat az IBM-mel a májusi Ifabón

Nem volt ez másképp az idei budapesti Ifabón sem. A standok között sétálgatva azt tapasztaltuk, hogy szinte mindegyik magára valamit is adó cég megpróbált olyasmivel előrukkolni, ami a hálózatok világával kapcsolatos.

Az IBM is erre helyezte a hangsúlyt. Ezúttal ugyan elmaradtak a tavalyi közönségcsalogató bemutatók, az érdeklődés mégsem csökkent, csak látványosan szakmaibb jellegű volt, és ez így is van rendjén. Ha már hálózatok, akkor kézenfekvő, hogy az IBM az összes szerverplatformját fölvonultatta egymás mellett; így bárki közelebbről is megismerkedhetett a PC Serverrel, az RS/6000-rel, az AS/400-zal és a System 390-es szerverekkel. De igazi újdonságot is hoztak az Ifabóra: immár működés közben is meg lehetett nézni az IBM hálózati komputerét, a Network Stationt. A jövőben jelentős szerepet szánnak ennek a technológiának, amely ha tényleg elterjed, koncepcionálisan fogja megújítani nemcsak a számítástechnikát, hanem egészen biztosan a felhasználók sokaságának mindennapjait is.

És miért ne lenne ez így, hiszen a stratégiai szövetségek már megköttettek. E téren nem titok, hogy míg a Microsoft, az Intell és még néhányan az „okos", egyre gyorsabb processzorokkal ellátott berendezéseket szeretnék a piacon meghonosítani, addig az IBM az Oracle-lal és a Sunnal inkább az egyszerűbb megoldások mellett tette le a voksát, amelyek, ha mást nem is veszünk számításba, már csak alacsony áruk miatt is előnyös pozícióból rajtolnak a piacon. Egyébként a Network Station terminálként eléri az s/390-es nagyszámítógépeket, az AS/400-as szerverek 5250 termináljaként is működik, az RS/6000-esekhez pedig a unixos világban szokásos ASCI, illetve X terminálként kapcsolható. Az új technológia igazi előnye, hogy a megbízható üzemeltetés igen egyszerűen megvalósítható, hiszen a gép minden alkalmazást és adatot a hálózati szerverről ér el, a megszokott PC-s programok mellett lehetőséget nyújt a Weben történő keresésre, és Java alkalmazásokat futtathat. A nagygépek és a hálózati masinák mellett persze látható volt egy-két hétköznapibb újdonság is, például az a TFT-s színes monitor, amelyik a képminőségen kívül leginkább kb. három centiméteres vastagságával nyűgözte le a nézőket. A hálózati számítástechnika szoftveroldalára is nagy súlyt fektetett az IBM. A Lotus például nagyon komoly megoldáshalmazt jelent a különféle alkalmazások hálózatról — ezen belül is Internetről és intranetekről — történő elérése szempontjából, a Tivoli pedig egy rendkívül komplex rendszermenedzsment megoldás.

Aki rászánt egy kis időt, jó néhány előadáson mélyíthette el a hálózati alkalmazásokkal kapcsolatos ismereteit is. Ezek közül is kiemelendő a hálózati kereskedelem, mivel nem kell jósnak lenni ahhoz, hogy az ember tudja, ez az a terület, amibe nem is olyan hosszú idő múlva lépten-nyomon bele fogunk botlani. A gyártók pedig nem véletlenül hangsúlyozzák minden fórumon, hogy mennyire biztonságosak azok a megoldások, amelyeket éppen piacra dobtak. Az IBM ún. Net-Commerce rendszere például már tartalmazza a Secure Electronic Transaction protokollt, amely biztonságos megoldást jelent a hálózati kereskedelemben arra, hogy megszokott hitelkártyánk segítségével vásárolhassunk.

A kereskedelmet megcélzó megoldásokon túl *Kiss Tibor*, az IBM Magyarországi Kft. szoftverüzletág-igazgatója még számos más újdonságra is felhívta a figyelmünket, amelyek az élet más területein hozhatnak forradalmi változásokat. A Digital Library, mint a neve is mutatja, lehetővé teszi, hogy óriási adatbázisokban kereshessen bárki, aki már rendelkezik egy egyszerű Internetböngészővel. Kell-e kézzelfoghatóbb bizonyíték arra, hogy az IBM nemcsak beszél az újdonságokról, hanem meg is valósítja azokat, mint az, hogy ez a szoftver is megtalálható már a világhálón, és az érdeklődő pillanatok alatt eljuthat a segítségével egy promóciós anyagig vagy egy videofilmig.

Az Internet kapcsán egyre többet hallani a világháló és az oktatás kapcsolatáról. Az IBM a kifejezetten iskolák számára kifejlesztett Net Vista szoftverével éppen ezt a világot szeretné érthetőbbé varázsolni a ma még iskolapadokat koptató jövőbeni felhasználói számára. A szakmának pedig ennél is érdekesebbek lehetnek azok a fejlesztőeszközök — mint például a VisualAge for Java —, amelyek segítségével a hálózatokra lehet újabb alkalmazásokat készíteni.

Ebből a rövid beszámolóból is látszik, hogy a számítástechnika világában immár jó ideje a hálózatoké a jövő. Így aztán nem is szólhatott másról az IBM idei jelmondata, mint hogy „Let's Net Together".

FERENCZI ZOLTÁN